UNIVERSIDADE DA CORUÑA

**ESCUELA UNIVERSITARIA POLITÉCNICA**

**Grado en Ingeniería Electrónica Industrial y Automática**

# TRABAJO DE FIN DE GRADO

TFG Nº: **770G01A123**

TÍTULO: **CONTROL REMOTO SOBRE LA PLANTA PEQUEÑA DE CONTROL DE NIVEL DEL LABORATORIO**

AUTOR: **DANIEL MÉNDEZ BUSTO**

TUTOR: **JOSÉ LUIS CALVO ROLLE**
**OSCAR FONTENLA ROMERO**

FECHA: **JUNIO DE 2017**

Fdo.: EL AUTOR

Fdo.: EL TUTOR

TÍTULO: **CONTROL REMOTO SOBRE LA PLANTA PEQUEÑA DE CONTROL DE NIVEL DEL LABORATORIO**

---

# ÍNDICE GENERAL

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBREIRO, S/N**

**15405 - FERROL**

FECHA: JUNIO DE 2017

AUTOR: EL ALUMNO

Fdo.: DANIEL MÉNDEZ BUSTO

# Índice de figuras

# Índice de tablas

# Listado de códigos de programación

TÍTULO: **CONTROL REMOTO SOBRE LA PLANTA PEQUEÑA DE CONTROL DE NIVEL DEL LABORATORIO**

---

# MEMORIA

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBREIRO, S/N**

**15405 - FERROL**

FECHA: JUNIO DE 2017

AUTOR: EL ALUMNO

Fdo.: DANIEL MÉNDEZ BUSTO

# Índice del documento MEMORIA

# 1 Objeto

El objetivo principal del presente trabajo de fin de grado es llevar a cabo el control remoto de la planta de nivel del laboratorio de optimización y control de la Escuela Universitaria Politécnica de Ferrol haciendo uso del entorno EASY JAVA SIMULATIONS y MATLAB para su posterior aplicación en docencia. Un usuario puede realizar el control de esta planta desde cualquier parte del mundo a través de internet.

# 2 Alcance

Los pasos que se llevan a cabo para la realización de este proyecto son los siguientes:

- Pruebas simples con la planta de nivel pequeña del laboratorio de optimización y control.
- Enlazar vía internet el programa de desarrollo EASY JAVA SIMULATIONS y el programa de cálculo MATLAB.
- Realizar la programación de la interface en EASY JAVA SIMULATIONS.
- Pruebas y validación de la propuesta sobre la planta pequeña de nivel del laboratorio.
- Análisis de resultados obtenidos.
- Elaboración de presupuesto de la aplicación desarrollada.

# 3 Antecedentes

En los últimos años, el campo de control y de automatización industrial experimenta un rápido crecimiento para favorecer el desarrollo de las actividades industriales. El sistema industrial tiende a una descentralización de sus elementos y toma gran importancia el mundo de las comunicaciones industriales y del control remoto. Esta evolución también afecta al mundo educativo, ya que gracias al control remoto en los laboratorios, el alumno no tienen que asistir *insitu* a las prácticas llevadas a cabo en estos laboratorios.

El laboratorio de optimización y control de la Escuela Universitaria Politécnica de Ferrol está dotado con una planta de nivel, en donde los alumnos diseñan y prueban diferentes tipos de reguladores. En la actualidad, los alumnos deben desplazarse al laboratorio para proceder al manejo de la planta, ya que ésta no cuenta con control remoto.

## 3.1. Planta de nivel del laboratorio

El funcionamiento de esta planta se basa en mantener un nivel de agua determinado en un depósito a partir de otro.

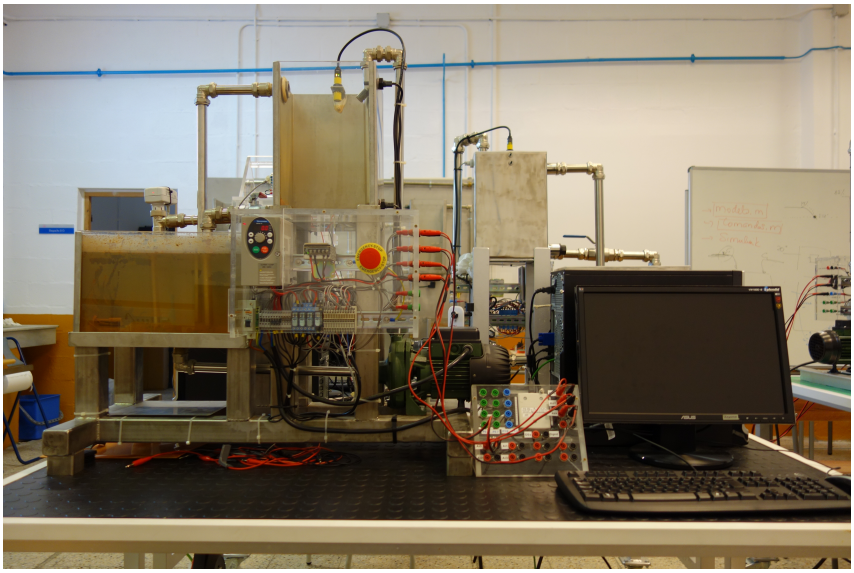


**Figura 3.1.0.1 –** Planta de nivel del laboratorio

Cómo se muestra en la imagen 3.1.0.1, la planta de nivel de laboratorio consta de dos depósitos, uno situado en la parte superior y en la parte inferior. El depósito de la parte superior es dónde se mantiene un determinado nivel de líquido, mientras que el depósito inferior es el

que lo alimenta. Para proteger la bomba centrífuga y para evitar que el depósito inferior se quede sin agua, éste es de mayor tamaño que el depósito superior.

Para alimentar el depósito superior, se cuenta de una bomba centrífuga. El vaciado del mismo se puede realizar por tres métodos. A través del tubo superior si el nivel del líquido es igual a la capacidad del depósito. A través de las válvulas que tiene el sistema, una válvula manual y una electro-válvula y mediante la propia bomba.

El régimen de giro de la bomba lo determina un variador de velocidad situado bajo del depósito superior. Este variador de velocidad es controlado mediante el sistema de control desarrollado en el software Matlab. Para poder comunicar el variador de velocidad con el ordenador es necesario hacer uso de una tarjeta de adquisición de datos. Gracias a ella, el ordenador obtiene el valor del nivel aportado por el sensor para medir el nivel (situado en la parte alta del depósito superior) y emite las órdenes de control al variador de velocidad.

En las siguientes subsecciones se detallan los elementos principales que forman la planta de nivel.

### 3.1.1. Bomba centrífuga

La bomba centrífuga que se encarga de elevar el líquido hacia el depósito superior es el modelo KST de SACI (figura 3.1.1.1).

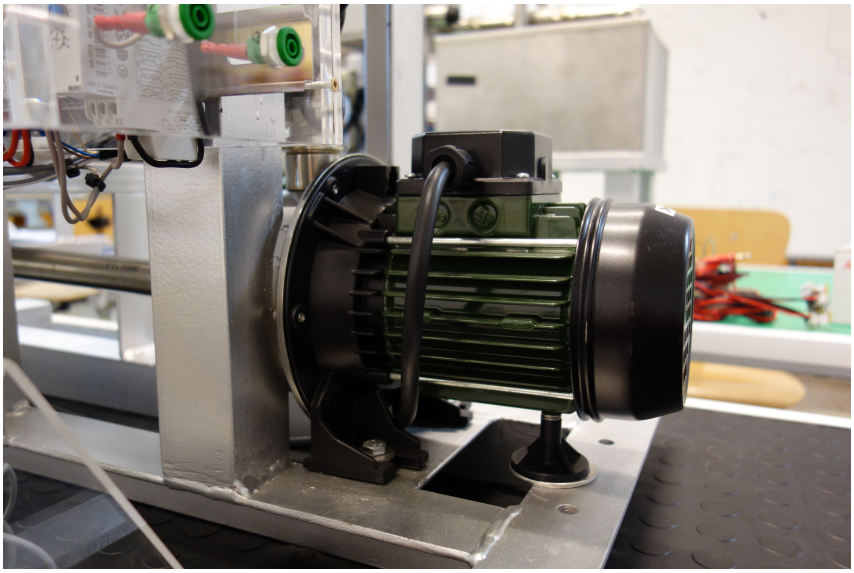


**Figura 3.1.1.1 –** Bomba Centrífuga

Las características más representativas de esta bomba son las siguientes:

- Tensión de alimentación trifásica entre 230 y 400 V.
- Frecuencia de alimentación de 50 Hz.
- Potencia nominal de 370 W.
- Caudal máximo de 6 $m^3/h$.

- Altura máxima de aspiración de 21,5 m.
- Régimen de trabajo de 2800 rpm.

### 3.1.2. Variador de velocidad

El control de la bomba se realiza mediante el variador de velocidad Altivor 31 de la casa Telemecanique Schneider Electric.

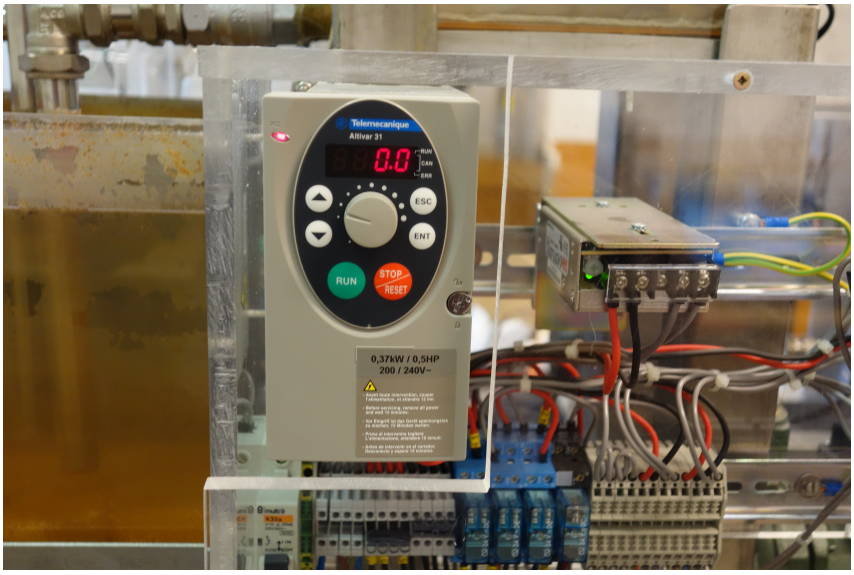



**Figura 3.1.2.1 –** Variador de Velocidad

Las características más representativas que presenta este variador son las siguientes:

- Tensión de entrada de 0 a 10 V.
- Salida trifásica con una tensión de 0 a 230 V.
- Una frecuencia de salida de 0 a 50 Hz.

El enlace entre el variador de velocidad y la bomba centrífuga es directo, ya que las características de salida del variador de velocidad y los parámetros de alimentación de la bomba centrífuga son compatibles

### 3.1.3. Tarjeta de Adquisición de datos

La tarjeta de adquisición de datos (DAQ NI USB-6008) es la encargada de transmitir las órdenes de control emitidas desde el ordenador al variador de velocidad. También, envía las medidas tomadas por el sensor de ultrasonidos al ordenador. La comunicación entre la tarjeta de adquisición de datos y el ordenador se realiza mediante cable USB. La tarjeta de adquisición de datos se muestra en la figura 3.1.3.1.

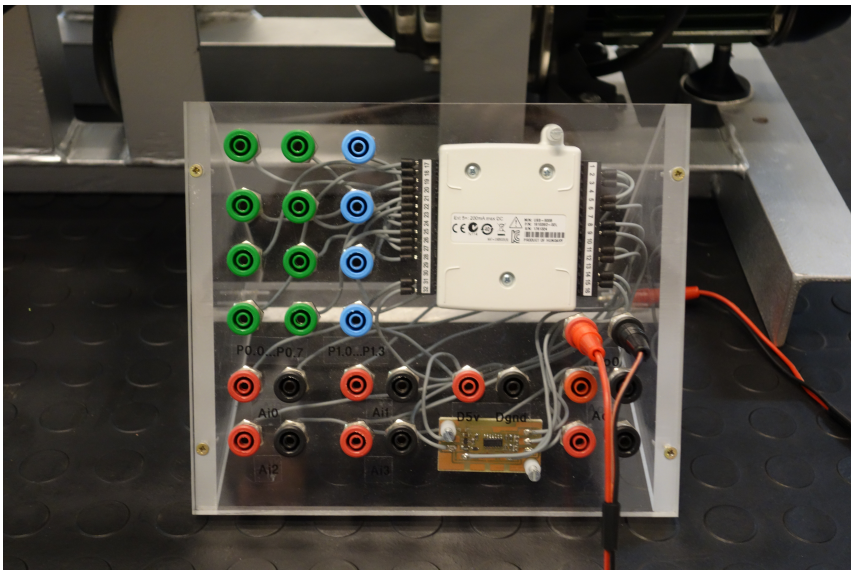


**Figura 3.1.3.1 –** Tarjeta de adquisición de datos

El modelo DAQ NI USB-6008 utiliza el siguiente protocolo de comunicación para enlazarse con el ordenador.

- **Inicialización de la DAQ:** Para comenzar la transferencia de datos, se debe de inicializar la tarjeta de adquisición. Se hace uso del Script *DAQ_START* (11.1).

- **Orden de escritura a la DAQ:** La escritura de un dato en la tarjeta de adquisición se se realiza mediante el Script *DAQ_Write()* (11.2).

- **Orden de lectura de la DAQ:** La lectura de un dato de la tarjeta de adquisición se realiza con el Script *DAQ_Read()* (11.3).

- **Desconexión de la DAQ:** Para cesar las comunicaciones entre la tarjeta de adquisición y el ordenador se hace uso del Script *DAQ_Stop* (11.4).

### 3.1.4. Sensor de ultrasonidos

El sensor de ultrasonidos es el encargado de realizar la medición del nivel de líquido en el depósito superior en tiempo real. El sensor que se utiliza es el modelo S18UAA de la casa Banner Enginnering (3.1.4.1).

Este sensor emite ondas de ultrasonidos que son recibidas por él mismo al rebotar contra el líquido a medir. Para conocer el nivel de líquido, mide el tiempo transcurrido entre la emisión y la recepción de la onda.

Se configura el mínimo rango de salida del sensor en la parte inferior del depósito (depósito vacío) y el máximo rango de salida en la parte superior del depósito (depósito lleno). La medida del nivel en el depósito superior se realiza porcentualmente.


**Figura 3.1.4.1 –** Sensor de ultrasonidos

Las características más representativas de este sensor son las siguientes:

- Tensión de entrada entre 10 y 30 V.
- Tensión de salida entre 0 y 10 V.

# 4 Normas y referencias

## 4.1. Disposiciones legales y normas aplicadas

En la ejecución del presente trabajo se cumple con la normativa de Trabajos Fin de Grado de la Escuela Universitaria Politécnica de Ferrol, aplicable a las titulaciones de Grado en Ingeniería Electrónica Industrial Y Automática y Grado en Ingeniería Eléctrica.

## 4.2. Bibliografía

[1] BASIL M., AL-HADITHI; *Análisis y Diseño de sistemas discretos de control*, Visión Net.

[2] CALVO ROLLE, JOSÉ LUIS; *Ajuste empírico de reguladores PID*.

[3] GARCÍA NOVO, FRANCISCO; *Estudio de métodos de detección de fallos para sistemas SISO*.

[4] CALVO ROLLE, JOSÉ LUIS; *Discretización de Sistemas Continuos*.

[5] CASTELEIRO ROCA, JOSÉ LUIS; *Scripts de comunicación de la tarjeta de adquisición de datos NI USB-6008 en Matlab*.

## 4.3. Programas utilizados

Los programas utilizados para la realización del presente trabajo son los siguientes:

- Matlab R2016b 32 bits
- Easy Java Simulations
- Microsoft Excel 2016
- TeXstudio
- MiKTex
- YawCam

## 4.4. Otras referencias

[i] *EjsWiki: Todo sobre Easy Java Simulations*. Disponible en: http://www.um.es/fem/EjsWiki/Es/HomePage

# 5 Definiciones y abreviaturas

- **EJS:** Easy Java Simulation
- **Z&N:** Ziegler - Nichols
- **Z&N NO:** Ziegler - Nichols No Overshoot
- **Z&N SO:** Ziegler - Nichols Some Overshoot
- **T&L:** Tyreus - Luyben
- **Regulador P:** Regulador proporcional
- **Regulador PI:** Regulador proporcional - integral
- **Regulador PD:** Regulador proporcional - derivativo
- **Regulador PID:** Regulador proporcional - integral - derivativo
- **DAQ:** Tarjeta adquisición de datos
- **SP:** Set point
- **N:** Parámetro de ajuste

# 6 Requisitos de diseño

Los requisitos que debe de cumplir el presente Trabajo Fin de Grado son:

- Poder implementar un control sobre la planta de nivel desde cualquier parte del mundo a través de internet.
- Poder visualizar en tiempo real el estado de la planta.
- Graficar en tiempo real el nivel, el error, la señal de control y la consigna.
- Permitir la realización del método de Relay - Feedback.
- El tiempo de muestreo ha de ser constante.
- Se deben de almacenar los datos de las variables nivel, error, señal de control y consigna en una base de datos. Esta base de datos se debe de poder visualizar desde el ordenador local y desde el ordenador remoto.

# 7 Análisis de las soluciones

Con el objetivo de dotar la planta de nivel con control remoto se estudia la posibilidad de implementar un **escritorio remoto** o desarrollar una **aplicación de control** desarrollada con el software *Easy Java Simulations*.

El escritorio remoto permite la utilización de una sesión en un ordenador local desde un equipo remoto ubicado en otro lugar. Para lograr la comunicación entre los dos equipos es necesario conexión a internet. Cuando se inicia la sesión remota, el usuario puede ejecutar Matlab y por lo tanto realizar un control sobre la planta de nivel.

Una aplicación de control desarrollada con *Easy Java Simulations* permite la conexión a un ordenador local desde un equipo remoto. La comunicación entre estos dos equipos se realiza mediante internet haciendo uso del protocolo *http*. Cuando se inicia la aplicación, el usuario está ante una *interface* que se enlaza con el software Matlab del ordenador local. A través del Matlab puede realizarse el control de la planta de nivel.

La ventaja que ofrece el escritorio remoto es que el usuario se encuentra con la *interface* original de Matlab por lo que obtiene un control total y absoluto sobre la planta de nivel. A la vez, esto es una desventaja, ya que no se puede controlar la programación realizada por el usuario y por lo tanto se pueden dar problemas como el deterioro o la ruptura de la propia planta, borrado de archivos ajenos al usuario o cambios indeseados en la configuración del equipo.

La aplicación de control tiene la ventaja que actúa como intermediaria entre el ordenador remoto y el ordenador local. De esta manera, se filtran las acciones que el usuario desea realizar sobre la planta de laboratorio, protegiéndola así contra ordenes perjudiciales para su integridad. La desventaja que ofrece es que el usuario no se encuentra con la *interface* original de Matlab, si no que trabaja a través de la aplicación de control.

Tras realizar el estudio de ambas alternativas se ha decidido hacer uso de la aplicación de control creada con *Easy Java Simulations*. A continuación se describe como es el funcionamiento de este software gratuito de desarrollo de aplicaciones Java.

## 7.1. EASY JAVA SIMULATIONS

Easy Java Simulations *(EJS)* es un entorno de programación que permite realizar simulaciones. Una simulación intenta reproducir gráficamente un fenómeno natural que fluye por distintos estados. Cada uno de estos estados se puede definir por variables [i].

En *EJS* se pueden crear simulaciones de alto nivel con un nivel medio de programación. El

lenguaje utilizado para el desarrollo en esta plataforma es Java. Se pueden crear aplicaciones Java independientes, aplicaciones Java multiplataforma o *applets*. Esta variedad de exportación, permite que una aplicación qué se genera con Easy Java Simulations se puede ejecutar en distintas plataformas e incluso en un navegador web [i].

Es compatible con programas de software como Matlab, Simulink o Labview y con elementos hardware como puede ser arduino. La comunicación entre los elementos software y el *EJS* se realiza a través de protocolo *http* o *tcp*. Esto dota a Easy Java Simulations de una gran potencia, ya que permite aprovechar todos los recursos de los software compatibles y con el añadido de poder realizarlo de forma remota haciendo uso de uno de los protocolos de comunicación mencionados anteriormente.

Se compone de dos ventanas principales: La consola de *EJS* y la interfaz de desarrollo de *EJS*.

### 7.1.1. Consola de Easy Java Simulations

La consola de *EJS* es el punto de partida para ejecutar el entorno de desarrollo. En ella se muestran mensajes de salida y posibles mensajes de error de las simulaciones creadas con *EJS* y, ocasionalmente, mensajes de error del propio Easy Java Simulations. Esta consola también permite configurar opciones básicas y avanzadas.

Desde la consola de *EJS* se pueden ejecutar varias ventanas de interface. Si se cierra la consola se detiene la total ejecución de Easy Java Simulations.

La consola de *EJS* se puede visualizar en la imagen 7.1.1.1:

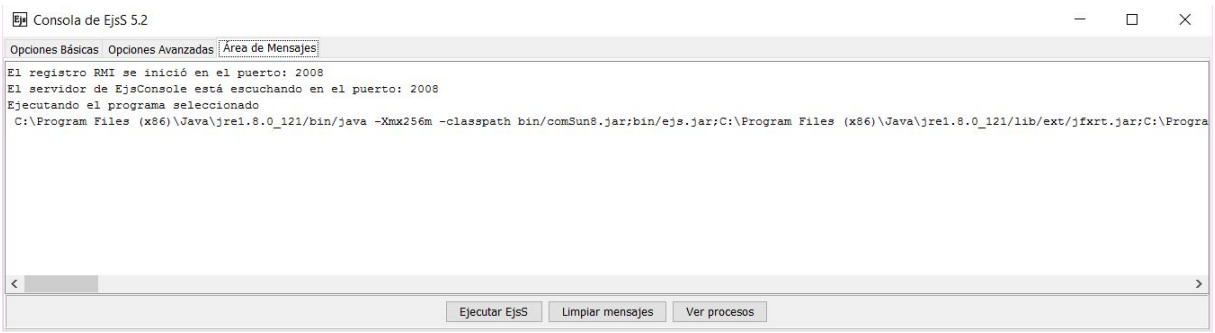



**Figura 7.1.1.1 –** Consola de Easy Java Simulations.

### 7.1.2. *Interface* de desarrollo de Easy Java Simulations

La *interface* de usuario de *EJS* es el entorno de programación de las aplicaciones. En la imagen 7.1.2.1 se observa la composición de la ventana principal.

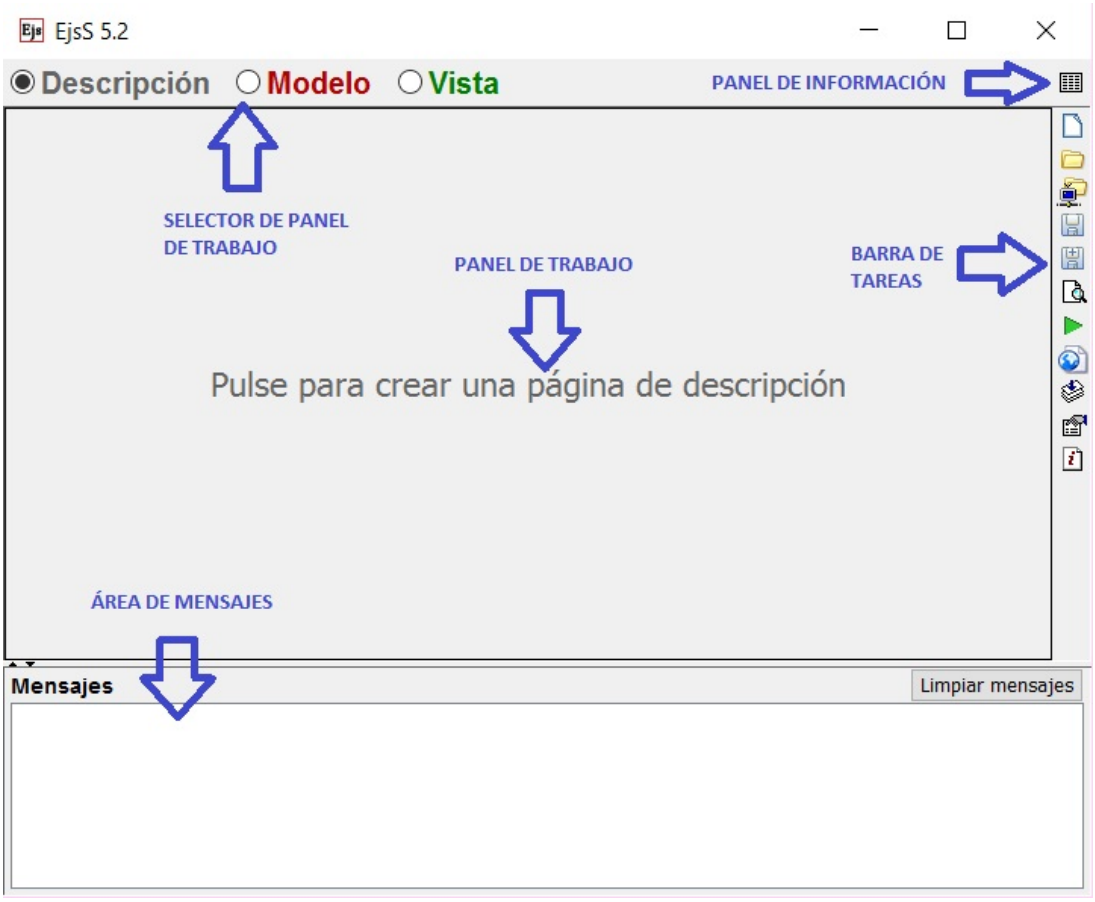



**Figura 7.1.2.1 –** *Interface* de usuario de Easy Java Simulations.

En las siguientes subsecciones se detallan los elementos que componen la *interface* de desarrollo de *Easy Java Simulations*.

#### 7.1.2.1. Selector de panel de trabajo

En el selector de panel de trabajo se permite navegar entre las pantallas de descripción, modelo y vista.

La pantalla de **descripción** (seleccionada en la figura 7.1.2.1) está destinada para realizar un resumen de la funcionalidad de la aplicación desarrollada.

La pantalla de **modelo** (figura 7.1.2.2) está destinada para realizar el desarrollo de código. Esta pantalla se subdivide a su vez en las siguientes pantallas: variables, inicialización, evolución, relaciones fijas, propio y elementos.

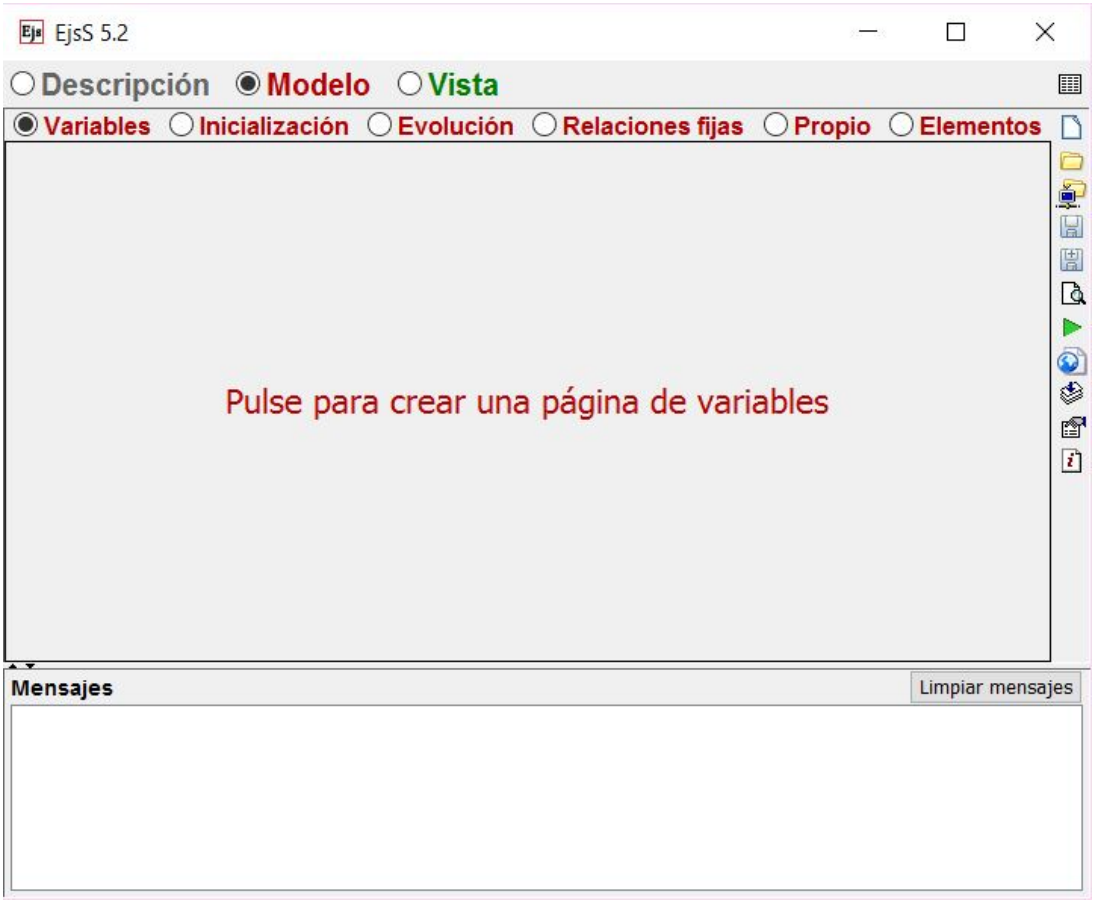



**Figura 7.1.2.2 –** Selector de panel de trabajo: Modelo

En la figura 7.1.2.2 se observa como está seleccionado el panel de variables. Si se realiza *click* sobre la pantalla central, aparece el cuadro mostrado en la figura 7.1.2.3.

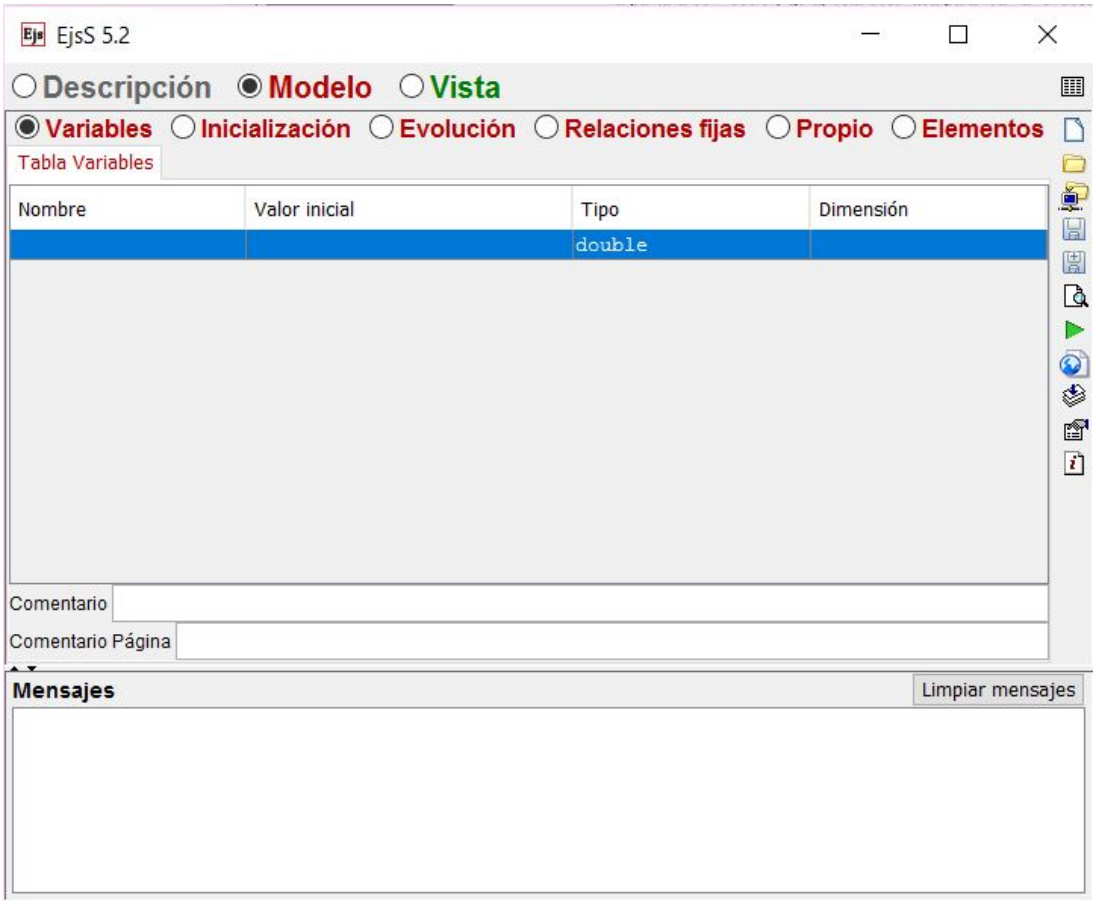



**Figura 7.1.2.3 –** Página de Modelo: Variables

Este cuadro está designado para la declaración de las variables de la aplicación. Para declarar una variable, se le designa un nombre, un valor inicial, un tipo (doble, entero, booleano o string) y una dimensión (único elemento o vector).

La pantalla de inicialización, representada en la figura 7.1.2.4, está destinada para programar aquellas instancias que se ejecutan al iniciar la aplicación. Estas sentencias se programan en lenguaje Java.

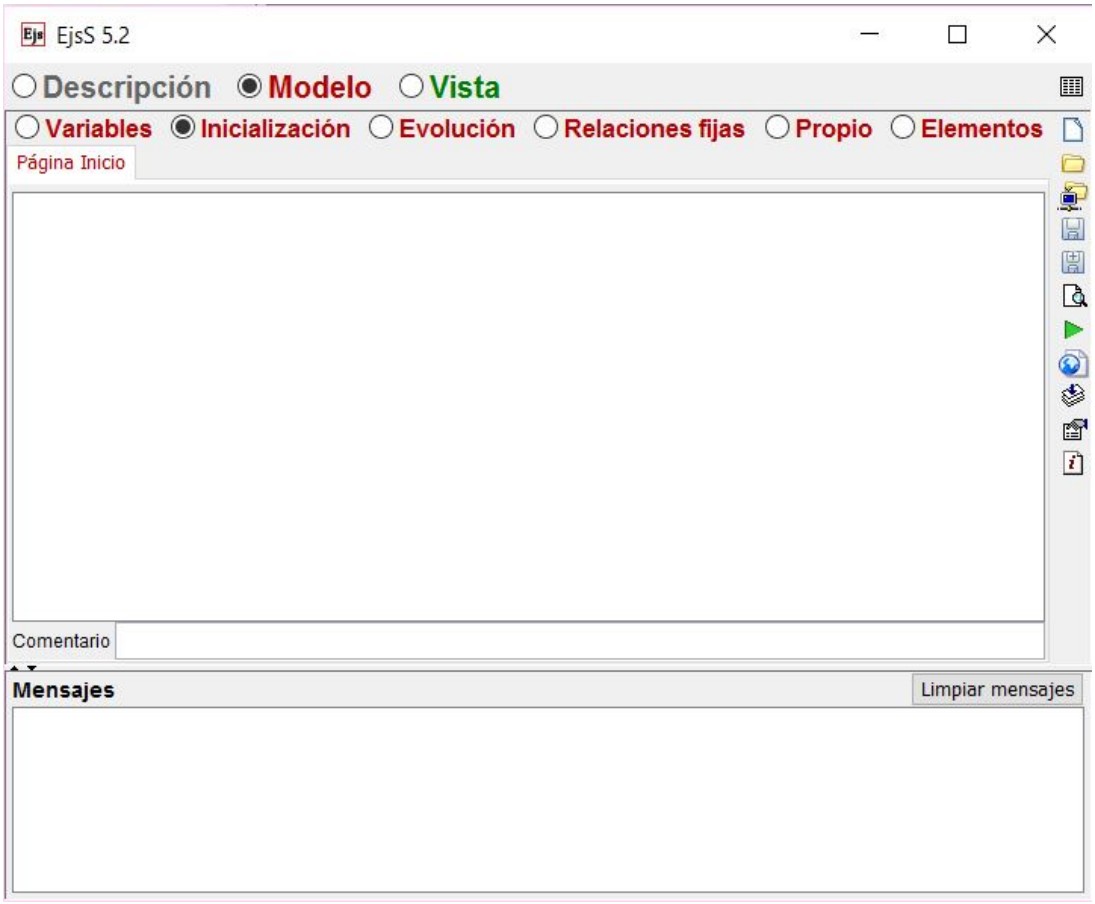



**Figura 7.1.2.4 –** Página de Modelo: Inicialización

La pantalla de evolución, representada en la figura 7.1.2.5, está destinada para desarrollar aquellas sentencias que muestran una evolución en el tiempo. Se puede elegir entre crear una página de código o crear una página de ecuaciones diferenciales ordinarias (EDO). En Easy Java Simulations, se pueden crear simulaciones de sistemas discretos o de sistemas continuos. Si se desea crear una simulación de un sistema discreto, se debe de seleccionar la opción de crear una página de código. Si por el contrario se desea realizar una simulación continua se puede seleccionar la opción de desarrollar el sistema mediante código Java (página de código) o la opción de introducir la ecuación en diferencias que describe el comportamiento del sistema (página de EDO) [i].

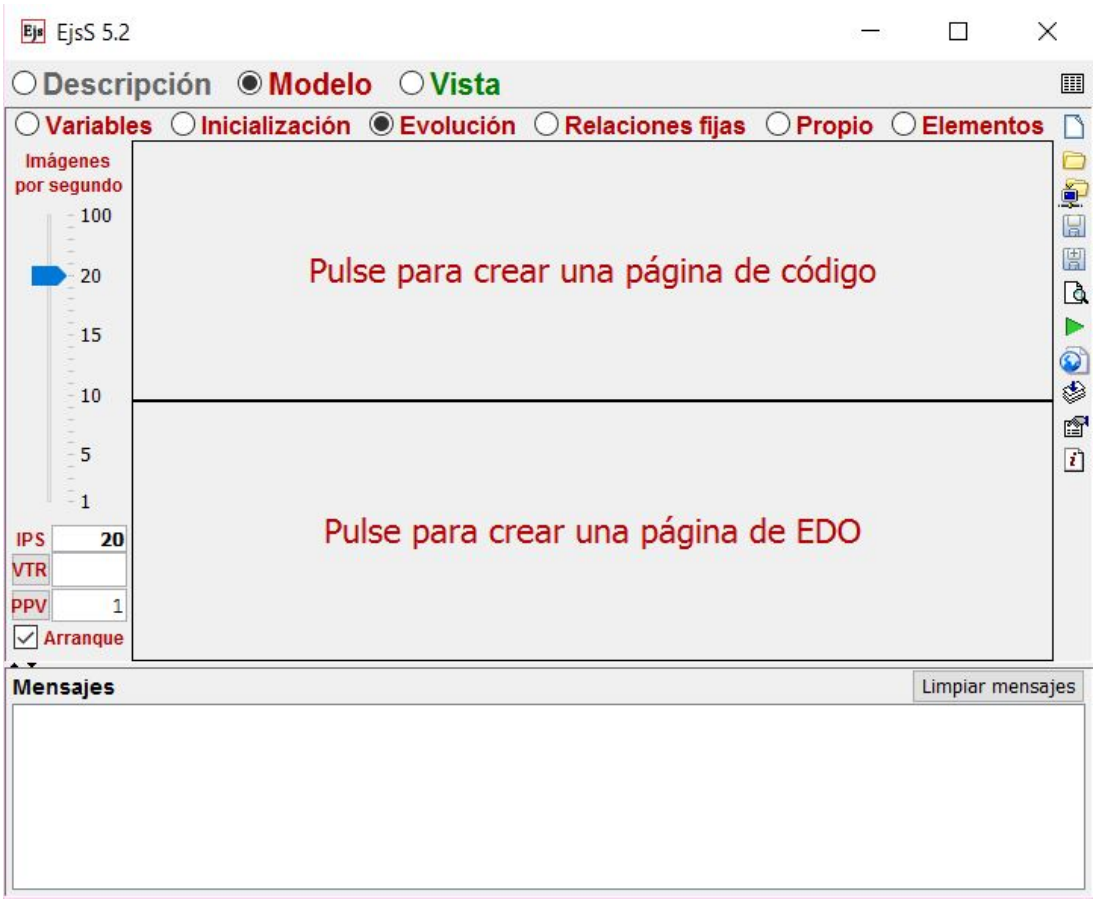



**Figura 7.1.2.5 –** Página de Modelo: Evolución

En la parte izquierda de la figura 7.1.2.5, se encuentra un menú en donde se puede modificar el número de imágenes por segundo (IPS), seleccionar una variable de tiempo real (VTR), modificar los pasos por visualización (PPV) y seleccionar el arranque.

El número de imágenes por segundo indica el número de veces que se representa la simulación por segundo. Se puede modificar deslizando el cursor por el deslizador, introduciendo un valor numérico en el cuadro *IPS* o asignando una variable o un valor en el campo *VTR*.

Los pasos por visualización especifican cuantas veces avanza el modelo entre representación y representación de la simulación.

La casilla de selección de arranque permite seleccionar si se desea que el apartado de

evolución comience con el inicio de la aplicación (seleccionado) o que se le dará la orden de inicio mediante código de programación (no seleccionado).

La pantalla de relaciones fijas, representada en la imagen 7.1.2.6, está destinada para programar la relación directa que presentan dos o más variables.

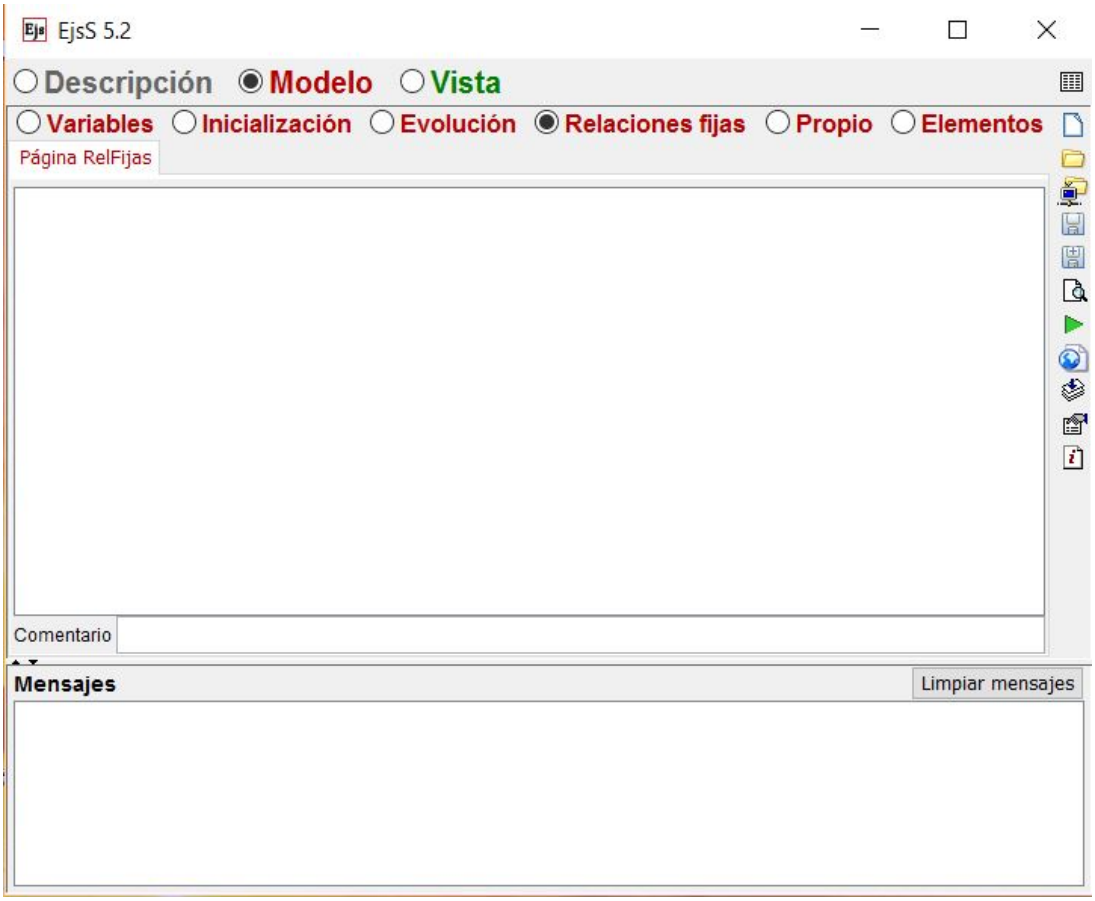



**Figura 7.1.2.6 –** Página de Modelo: Relaciones fijas

La pantalla de propio, representada en la figura 7.1.2.7, está destinada para la programación de funciones en Java. Los parámetros que definen estas funciones son el nombre, las variables de entrada y las variables de salida. El nombre será el identificador para proceder a la llamada de la función. Las variables de entrada, que puede haberlas o no, son parámetros que utiliza la función en su ejecución y son transferidas en el momento de la llamada. Las variables de salida, que puede haberlas o no, son aquellas variables en las que la función vuelca un resultado al finalizar su ejecución [i].

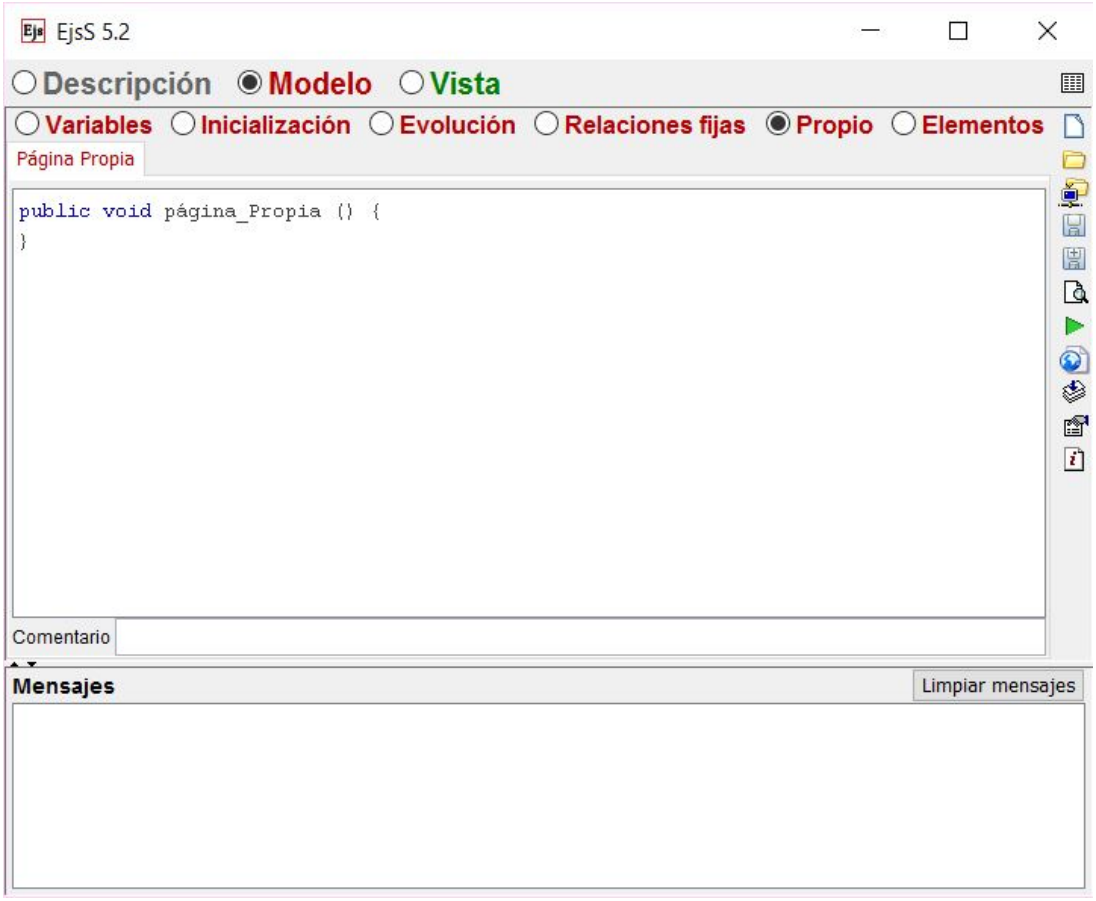



**Figura 7.1.2.7 –** Página de Modelo: Propio

La pantalla de elementos, representada en la figura 7.1.2.8, está destinada para configurar los elementos software y/o hardware que se pueden vincular con Easy Java Simulations.

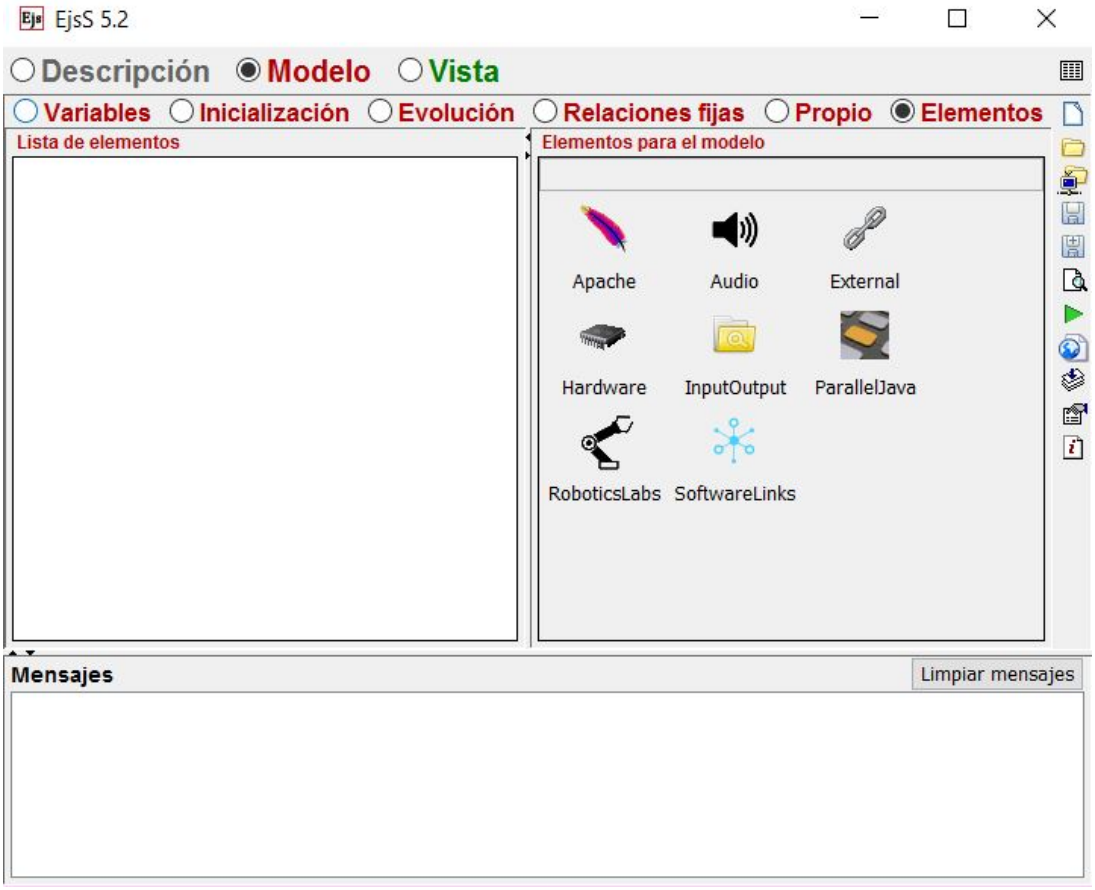



**Figura 7.1.2.8 –** Página de Modelo: Elementos

En la pantalla de **vista** se diseña el aspecto gráfico de la aplicación. La programación de esta ventana consiste en vincular objetos a variables propias del proceso.
La ventana principal de diseño se compone del árbol de elementos y de los elementos para la vista.
En el interior del árbol de elementos, se encuentra la vista de la simulación. Todos los elementos de la *interface* gráfica se cuelgan de ella y se estructuran en forma de árbol.En los elementos de la vista de encuentran los elementos de *interface*, los elementos de dibujo 2D y los elementos de dibujo 3D. En estos menús se seleccionan los objetos gráficos de la visualización.
En la parte inferior se puede observar una casilla de selección que permite ver la vista previa de la simulación [i].
En la imagen 7.1.2.9 se muestra el aspecto de la ventana de vista.

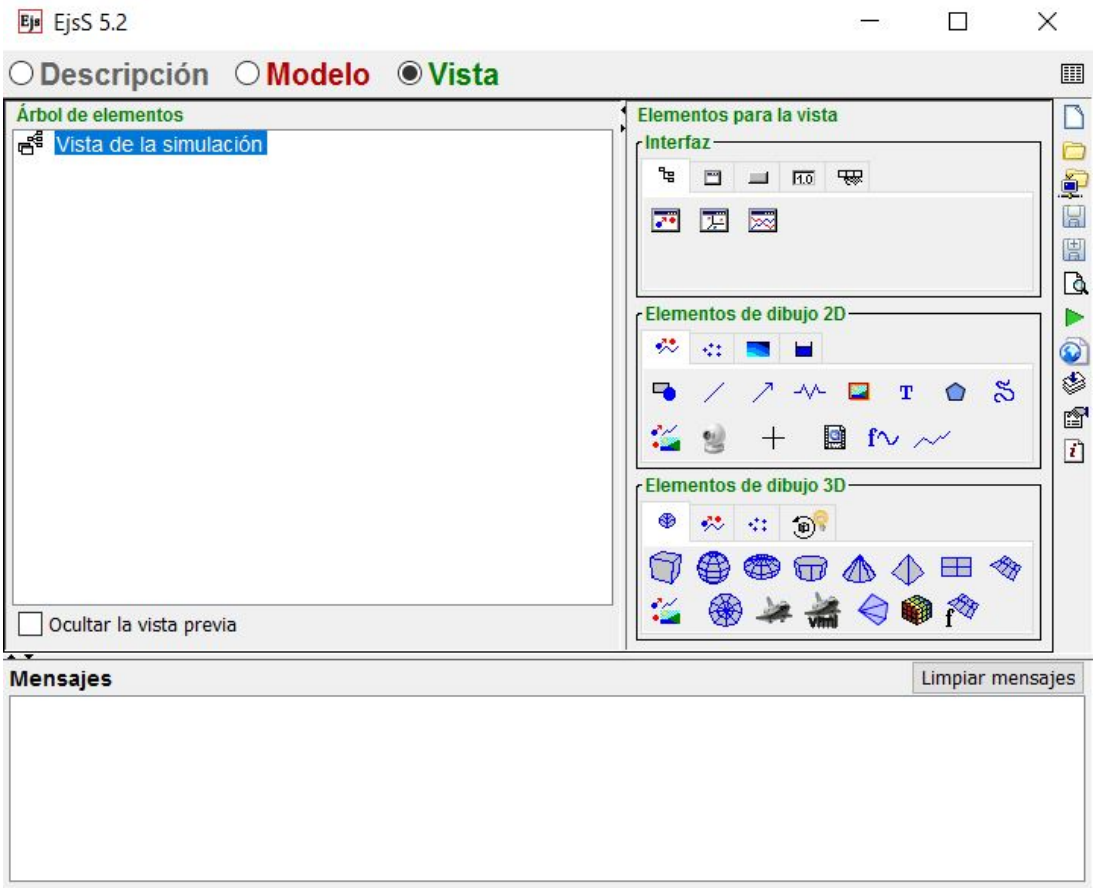



**Figura 7.1.2.9 –** Página de Vista

En la siguiente tabla se muestran los elementos de interface [i].

| Símbolo | Nombre | Descripción |
|---|---|---|
| | Drawing Frame | Ventana de ejemplo compuesta por elementos 2D |
| | Drawing Frame 3D | Ventana de ejemplo compuesta por elementos 3D |
| | Plotting Frame | Ventana de ejemplo de gráficos |
| | Ventana | Ventana vacía para elementos 2D |
| | Ventana de diálogo | Ventana vacía y fija para elementos 2D |
| | Panel | Panel contenedor básico |
| | Panel desplazable | Panel contenedor con propiedades de desplazamiento |

| Símbolo | Nombre | Descripción |
|---|---|---|
|  | Panel dividido | Panel contenedor con dos áreas separadas |
|  | Panel separador | Panel contenedor con separadores |
|  | Barra de herramientas | Un contenedor en forma de barra de herramientas |
|  | Panel de dibujo | Un contenedor 2D para elementos de dibujo |
|  | Panel con ejes | Un contenedor 2D con ejes |
|  | Panel de dibujo 3D | Un contenedor 3D |
|  | Etiqueta | Una etiqueta decorativa |
|  | Ecuación | Campo destinado para una ecuación matemática |
|  | Botón | Un botón para acciones |
|  | Botón dos estados | Un botón para acciones de dos estados |
|  | Selector | Un selector para varios booleanos |
|  | Botón radio | Un botón para varios booleanos |
|  | Selector con sonido | Un selector con sonido |
|  | Separador | Una línea de separación |
|  | Barra | Una barra que visualiza un valor |
|  | Deslizador | Una barra que visualiza o modifica un valor |
|  | Mando | Mando circular para modificar un valor |
|  | Tabla de datos | Una tabla de datos |
|  | Lista | Lista desplegable para selección de texto |
|  | Campo numérico | Se visualiza o se modifica un dato numérico |
|  | Campo de texto | Se visualiza o se modifica un texto |

| Símbolo | Nombre | Descripción |
|---|---|---|
|  | Campo clave | Campo para introducir una palabra secreta |
|  | Función | Campo de texto para evaluar una función |
|  | Área de texto | Área que imprime un texto |
|  | Panel matriz | Panel para editar una matriz |
|  | Barra menú | Una barra de menú |
|  | Menú | Un menú desplegable |
|  | Botón menú | Un botón para menús |
|  | Selector menú | Un selector para menús |
|  | Botón radio menú | Un botón radio para menús |
|  | Separador menú | Un separador para menús |
|  | Forma | Una forma 2D (elipse,cuadrado) |
|  | Segmento | Un segmento 2D |
|  | Flecha | Una flecha 2D |
|  | Muelle | Un muelle 2D |
|  | Imagen | Una imagen 2D |
|  | Texto | Un texto 2D |
|  | Polígono | Un polígono 2D |
|  | Rastro | Un rastro 2D |
|  | Grupo | Un grupo 2D |
|  | Basic Web Cam | Una imagen básica de Web Cam |
|  | Cursor | Un cursor para un panel 2D |
|  | Video | Un clip de video |

| Símbolo | Nombre | Descripción |
|---|---|---|
|  | Curva analítica | Una curva en 2D dada por expresiones analíticas |
|  | Traza | Una secuencia de puntos |
|  | Conjunto de formas | Un conjunto de formas 2D |
|  | Conjunto de segmentos | Un conjunto de segmentos 2D |
|  | Conjunto de flechas | Un conjunto de flechas 2D |
|  | Conjunto de muelles | Un conjunto de muelles 2D |
|  | Conjunto de imágenes | Un conjunto de imágenes 2D |
|  | Conjunto de textos | Un conjunto de textos 2D |
|  | Conjunto de polígonos | Un conjunto de polígonos 2D |
|  | Conjunto de rastros | Un conjunto de rastros 2D |
|  | Conjunto de trazas | Un conjunto de trazas 2D |
|  | Retícula | Visualización de un conjunto de 0 y 1 |
|  | Retícula Células | Visualización tipo tablero de un campo escalar |
|  | Retícula puntos | Visualización de un conjunto de bytes |
|  | Nube | Una nube de puntos |
|  | Histograma | Un histograma para datos |
|  | Barrido datos | Una visualización 2D de puntos con color |
|  | Barrido bytes | Una visualización 2D de bytes con color |
|  | Datos Complejos | Un conjunto de números complejos |
|  | Retícula triangular Bytes | Visualización triangular para conjunto de 0 y 1 |
|  | Retícula triangular Bytes | Visualización triangular para conjunto de enteros |

| Símbolo | Nombre | Descripción |
|---|---|---|
|  | Campo vectores | Un campo vectores 2D |
|  | Campo vectores analítico | Un campo vectores analíticos 2D |
|  | Campo escalar | Un campo escalar 2D |
|  | Campo escalar analítico | Un campo escalar analítico 2D |
|  | Campo escalar complejo | Un campo escalar complejo 2D |
|  | Malla | Un campo sobre una malla delimitada |
|  | Tanque | Un tanque para líquido |
|  | Tubería | Una tubería interactiva |
|  | Válvula | Una válvula para líquidos |
|  | Bomba | Una bomba para líquidos |
|  | Símbolo | Un símbolo de control |
|  | Línea | Una línea de control |
|  | Caja 3D | Una caja 3D |
|  | Esfera 3D | Una esfera 3D |
|  | Elipsoide 3D | Una elipsoide 3D |
|  | Cilindro 3D | Un cilindro 3D |
|  | Cono 3D | Un cono 3D |
|  | Tetraedro 3D | Un tetraedro 3D |
|  | Plano 3D | Un plano 3D |
|  | Superficie 3D | Una superficie 3D |
|  | Grupo 3D | Un grupo 3D |

| Símbolo | Nombre | Descripción |
|---|---|---|
|  | Disco 3D | Un disco 3D |
|  | Objeto 3D | Un objeto 3D descrito por un fichero |
|  | Objeto VRML | Un objeto 3D descrito por un fichero VRML |
|  | Elosado 3D | Una superficie en 3D compuesta de losas poligonales |
|  | Malla 3D | Un campo sobre una malla en 3D delimitada |
|  | Superficie analítica 3D | Una superficie en 3D dada por expresiones analíticas |
|  | Partícula 3D | Una partícula en 3D |
|  | Segmento 3D | Un segmento en 3D |
|  | Flecha 3D | Una flecha en 3D |
|  | Muelle 3D | Un muelle en 3D |
|  | Imagen 3D | Una imagen en 3D |
|  | Texto 3D | Un texto en 3D |
|  | Polígono 3D | Un polígono en 3D |
|  | Traza 3D | Una secuencia de puntos en 3D |
|  | Curva analítica 3D | Una curva en 3D dada por una expresión analítica |
|  | Puntos 3D | Una nube de puntos 3D |
|  | Campo de vectores 3D | Un campo de vectores 3D |
|  | Campo vectorial analítico 3D | Un campo de vectores 3D de expresión analítica |
|  | Panel 2D en 3D | Un panel de dibujo 2D para una escena 3D |
|  | Conjunto cajas 3D | Un conjunto de cajas 3D |
|  | Conjunto esferas 3D | Un conjunto de esferas 3D |
|  | Conjunto elipsoides 3D | Un conjunto de elipsoides 3D |
|  | Conjunto cilindros 3D | Un conjunto de cilindros 3D |

| Símbolo | Nombre | Descripción |
|---|---|---|
|  | Conjunto conos 3D | Un conjunto de conos en 3D |
|  | Conjunto tetraedros 3D | Un conjunto de tetraedros en 3D |
|  | Conjunto planos 3D | Un conjunto de planos en 3D |
|  | Conjunto superficies 3D | Un conjunto de superficies en 3D |
|  | Conjunto partículas 3D | Un conjunto de partículas en 3D |
|  | Conjunto segmentos 3D | Un conjunto de segmentos en 3D |
|  | Conjunto flechas 3D | Un conjunto de flechas en 3D |
|  | Conjunto muelles 3D | Un conjunto de muelles en 3D |
|  | Conjunto imágenes 3D | Un conjunto de imágenes en 3D |
|  | Conjunto textos 3D | Un conjunto de textos en 3D |
|  | Conjunto polígonos 3D | Un conjunto de polígonos en 3D |
|  | Conjunto trazas 3D | Un conjunto de trazas en 3D |
|  | Rotación 3D | Una rotación para elementos 3D |
|  | Rotación X 3D | Una rotación alrededor del eje X en 3D |
|  | Rotación Y 3D | Una rotación alrededor del eje Y en 3D |
|  | Rotación Z 3D | Una rotación alrededor del eje Z en 3D |
|  | Cuaternión 3D | Una transformación 3D por cuartenión |
|  | Alineamiento 3D | Una rotación que alinea dos vectores 3D |
|  | Matriz 3D | Una transformación 3D basada en una matriz |
|  | Luz 3D | Una luz para una escena 3D |

**Tabla 7.1.2.1 –** Elementos de *interface*.

#### 7.1.2.2. Panel de trabajo

El panel de trabajo es la zona en dónde se realiza la programación de la aplicación. Para añadir una ventana de programación se debe de realizar click en cualquier punto del panel de trabajo. Al realizar esta acción aparece una ventana emergente pidiendo el nombre de la ventana a crear. Este panel se encuentra en todos los campos disponibles en el selector de panel de trabajo.

#### 7.1.2.3. Área de mensajes

El área de mensajes es la utilizada por Easy Java Simulations para transmitir errores o estados de la aplicación que se esta a desarrollar. Estos mensajes ayudan al usuario a corregir errores de compilación, informan si la aplicación se ha guardado correctamente, muestran el estado de la simulación ...

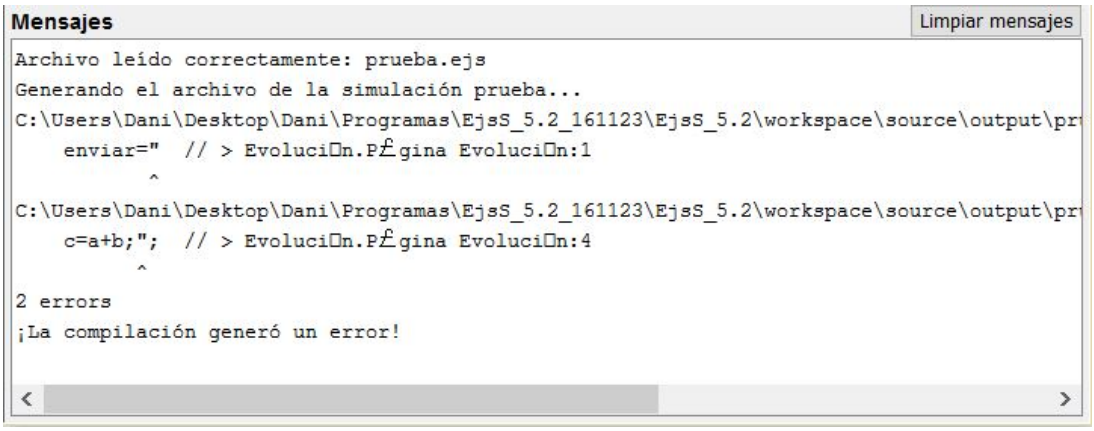



**Figura 7.1.2.10 –** Ejemplo área de mensajes

El área de mensajes presenta un botón nombrado como ”Limpiar mensajes”(se puede ver en la figura 7.1.2.10) y permite borrar los mensajes existentes en dicho área.

#### 7.1.2.4. Panel de información

El panel de información permite la configuración de la aplicación Easy Java Simulations. Cuando se accede a este menú se muestra la ventana mostrada en la figura 7.1.2.11.

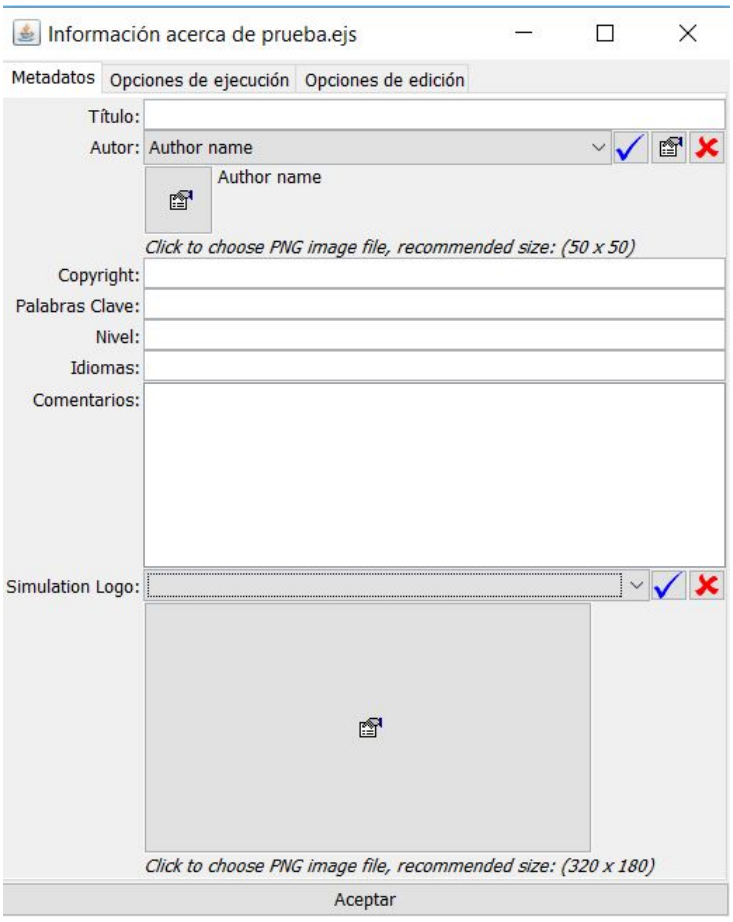



**Figura 7.1.2.11 –** Panel de información: Introducción de datos

En este panel se pueden configurar datos del autor e insertar una imagen para la aplicación a desarrollar.
Como se observa en la figura 7.1.2.11, en la parte superior se puede selecciona las opciones de ejecución o las opciones de edición. Estos ajustes vienen por defecto configurados y sólo sería necesarios ajustarlos en el caso de desarrollar aplicaciones muy complejas.

### 7.1.2.5. Barra de tareas

La barra de tareas contiene accesos directos para facilitar al usuario la programación de la aplicación. Desde ella se puede abrir un nuevo proyecto, guardar una aplicación en desarrollo, cambiarle el nombre a la aplicación que se está a desarrollar, ejecutar simulaciones o crear la aplicación en formato .jar.
Cómo se observa en la figura 7.1.2.1, la barra de tareas está situada en el lateral derecho.
En la siguiente tabla se muestran los componentes de este menú y se explica la función de cada uno.

| Símbolo | Descripción |
|---|---|
|  | Permite crear una aplicación nueva |
|  | Permite abrir una aplicación desarrollada en EJS |
|  | Permite añadir librerías de EJS |
|  | Guarda la aplicación que se está a desarrollar en la actualidad |
|  | Se puede renombrar la aplicación cuando se guarda |
|  | Permite la búsqueda de palabras |
|  | Ejecuta la aplicación en modo simulación |
|  | Traduce la aplicación a otro idioma |
|  | Crea la aplicación en formato .jar |
|  | Permite modificar opciones de la simulación |
|  | Abre la ayuda de EJS |

**Tabla 7.1.2.2 –** Elementos de la barra de tareas.

## 7.2. Regulación de la planta de nivel

El objetivo es que a través de la aplicación de control el usuario pueda realizar una regulación sobre el nivel del depósito superior de la planta de nivel.

En este proyecto se dota a la aplicación de control con reguladores PID y de sus simplificaciones. Se ha optado por esta opción, ya que los reguladores PID y sus simplificaciones se estudian en la materia de Ingeniería de Control en el tercer curso de Ingeniería Electrónica Industrial y Automática.

### 7.2.1. Reguladores

Un regulador es un algoritmo programado en un computador que se utiliza para corregir comportamientos no ideales de sistemas reales que están sometidos a perturbaciones, errores o características no lineales en su funcionamiento.

Las acciones que definen el tipo de regulador son la acción proporcional, la acción integral y la acción derivativa.

El regulador P está compuesto por la acción proporcional, el regulador PI por las acciones proporcional e integral, el regulador PD por las acciones proporcional y derivativa y el regulador PID por las tres acciones.

El regulador PID es el más utilizado en la actualidad, ya que su respuesta se adapta a la mayoría de los sistemas.

#### 7.2.1.1. Acción proporcional

Se encarga de modificar principalmente la ganancia del sistema. Se emplea para disminuir el error, sin embargo esto puede aumentar las oscilaciones. Esta acción modifica el régimen transitorio y permanente [1].

#### 7.2.1.2. Acción integral

Se encarga de anular el error en régimen permanente. El uso de esta acción aumenta en 1 el tipo del sistema, por lo que aumentará la precisión del mismo con el riesgo de inestabilizarlo. Esta acción introduce un polo en el origen, eliminando el error de posición. Modifica principalmente el régimen permanente [1].

#### 7.2.1.3. Acción derivativa

Se encarga de modificar el comportamiento transitorio con el fin de modificar la velocidad de respuesta del sistema y las oscilaciones del transitorio. Esta acción introduce un cero en el sistema. Sin embargo, presenta grandes inconvenientes, amplifica la señales de alta frecuencia (ruido) que pueda presentar la señal, además de producir saturaciones en los actuadores cuando la señal de error varía bruscamente [1].

### 7.2.2. Implementación de los reguladores

Las acciones características de los reguladores pueden presentar distintas morfologías. En el presente trabajo se resalta un regulador sin corrección y un regulador con corrección (parámetro N).

El **regulador sin corrección** discretizado se obtiene analíticamente mediante la resolución de la ecuación diferencial PID estándar [1].

$$\boxed{u\left(t\right) = K_p\left(e\left(t\right) + \frac{1}{T_i}\int e\left(t\right)dt + T_d\frac{de\left(t\right)}{dt}\right)} \tag{7.2.2.1}$$

Donde $u\left(t\right)$ es la señal de control expresada en tanto por ciento, $K_p$ es la constante proporcional,$e\left(t\right)$ es el error,$T_i$ el tiempo integral y $T_d$ el tiempo derivativo.

La ecuación en diferencias del algoritmo PID tomará la siguiente forma [1]:

$$\boxed{u\left(k\right) = K_p\left(e\left(k\right) + \frac{T}{T_i}\sum_{k=1}^{\infty}\frac{e\left(k-1\right)+e\left(k\right)}{2} + \frac{T_d}{T}\left(e\left(k\right)-e\left(k-1\right)\right)\right)} \tag{7.2.2.2}$$

Donde $u(k)$ es la señal de control expresada en tanto por ciento, $K_p$ es la constante proporcional, $e(k)$ es el error en el instante *k*, $e(k-1)$ es el error en el instante *k-1*, $T_i$ el tiempo integral y $T_d$ el tiempo derivativo [4].

El **regulador con corrección** rige su funcionamiento por las siguientes fórmulas:

$$e(k) = r(k) - y(k) \tag{7.2.2.3}$$

$$i(k) = C_1(e(k) + e(k-1)) + i(k-1) \tag{7.2.2.4}$$

$$d(k) = C_2(e(k) - e(k-1)) + C_3 d(k-1) \tag{7.2.2.5}$$

$$\boxed{u(k) = K * (e(k) + i(k) + d(k))} \tag{7.2.2.6}$$

Siendo $C_1$, $C_2$ y $C_3$:

$$C_1 = \frac{T_s}{2T_i} \tag{7.2.2.7}$$

$$C_2 = \frac{2T_d}{\frac{2T_d}{N} + T_s} \tag{7.2.2.8}$$

$$C_3 = \frac{\frac{2T_d}{N} - T_s}{\frac{2T_d}{N} + T_s} \tag{7.2.2.9}$$

Donde $N$ es un parámetro de ajuste de la acción derivativa.

#### 7.2.2.1. Regulador P

El regulador P es el controlador más sencillo que se puede implementar, ya que únicamente tiene parte proporcional y busca la regulación mediante la variación del parámetro K. El diagrama de bloques de este regulador se muestra en la siguiente imagen.

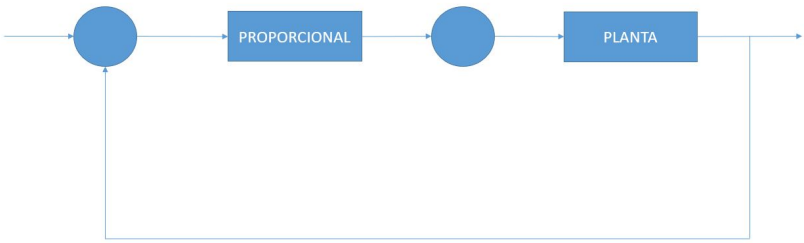



**Figura 7.2.2.1 –** Diagrama de bloques del regulador proporcional.

La ecuación que rige el funcionamiento de este regulador es la siguiente. Esta ecuación es común para el **regulador sin corrección** y para el **regulador con corrección** [1].

$$\boxed{u\left(k\right) = K_p\left(e\left(k\right)\right)} \tag{7.2.2.10}$$

### 7.2.2.2. Regulador PI

El regulador PI está compuesto por la acción proporcional y por la acción integral. La acción integral corrige el error de posición que se tiene con un regulador tipo P. A su vez, este introduce sobreoscilación y acerca el sistema a la inestabilidad. El diagrama de bloques de este regulador se muestra en la siguiente figura:

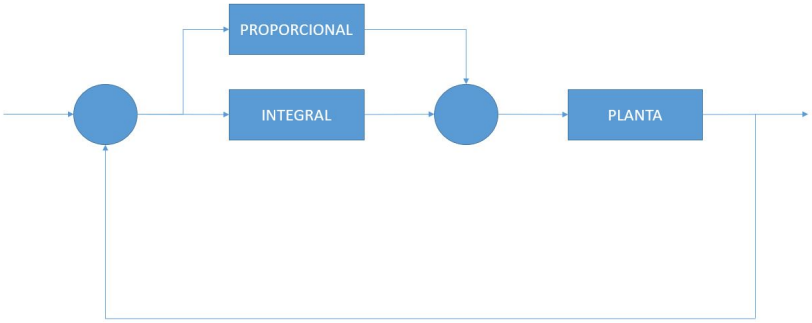



**Figura 7.2.2.2 –** Diagrama de bloques del regulador PI.

La ecuación que define el **regulador PI sin corrección** es la siguiente [1]:

$$\boxed{u\left(k\right) = K_p\left(e\left(k\right) + \frac{T}{T_i}\sum_{k=1}^{\infty}\frac{e\left(k-1\right)+e\left(k\right)}{2}\right)} \tag{7.2.2.11}$$

La ecuación que define el **regulador PI con corrección** es la siguiente [4]:

$$\boxed{u\left(k\right) = K * \left(e\left(k\right) + i\left(k\right)\right)} \tag{7.2.2.12}$$

#### 7.2.2.3. Regulador PD

El regulador PD está compuesto por la acción proporcional y la acción derivativa. El uso de este regulador no es muy habitual, ya que la acción derivativa es la más difícil de controlar. El diagrama de bloques de este regulador se muestra en la siguiente imagen:

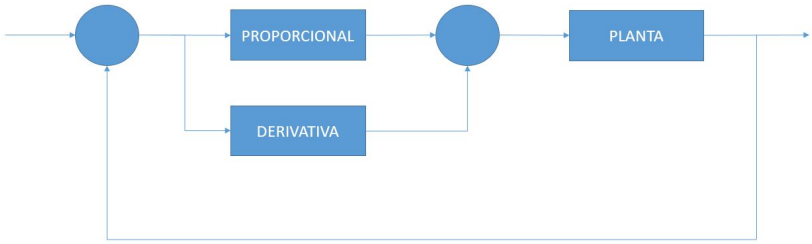



**Figura 7.2.2.3 –** Diagrama de bloques del regulador PD.

La ecuación que define el **regulador PD sin corrección** es la siguiente [1]:

$$\boxed{u(k) = K_p \left( e(k) + \frac{T_d}{T} (e(k) - e(k-1)) \right)} \tag{7.2.2.13}$$

La ecuación que define el **regulador PD con corrección** es la siguiente [4]:

$$\boxed{u(k) = K * (e(k) + d(k))} \tag{7.2.2.14}$$

#### 7.2.2.4. Regulador PID

El regulador PID está compuesto por la acción proporcional, la acción integral y la acción derivativa. La acción integral corrige el error de posición cometido por la acción proporcional, a su vez introduce sobreoscilación y acerca al sistema a la inestabilidad. La acción derivativa se encarga de mejorar los defectos introducidos por la acción integral. El diagrama de bloques de este regulador se muestra en la siguiente imagen:

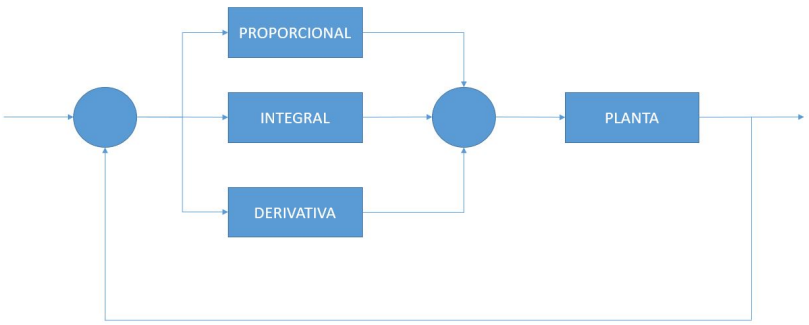



**Figura 7.2.2.4 –** Diagrama de bloques del regulador PID.

La ecuación 7.2.2.2 rige el funcionamiento del **regulador PID sin corrección** [1]. En el caso del **regulador PID con corrección** se hace uso de la ecuación 7.2.2.6 [4].

### 7.2.3. Método *Relay - Feedback*

Para hacer posible la implantación de un regulador es necesario calcular los parámetros propios del regulador $K_p$, $T_i$ y $T_d$. Estos parámetros se calculan a partir de las características del sistema a regular. El método del relay - feedback permite hallar la ganancia crítica ($K_c$) y el periodo de oscilación sostenida ($T_c$) del sistema [2].

El método del *relay - feedback* es un método empírico que consiste en llevar al sistema al estado de oscilación mediante el uso de un relé [2].

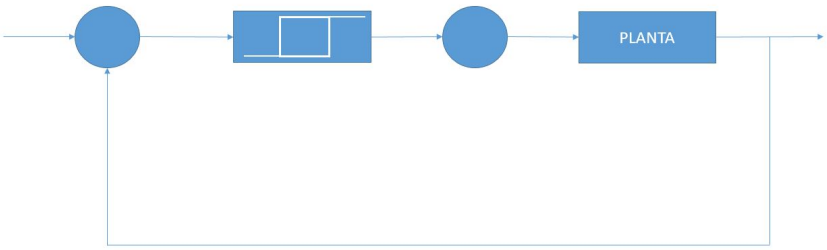



**Figura 7.2.3.1 –** Esquema realización Relay - Feedback.

El relé que se utiliza en el método empírico del *Relay - Feedback* ha de tener un parámetro de histéresis (h) y una amplitud (d) determinada [2].

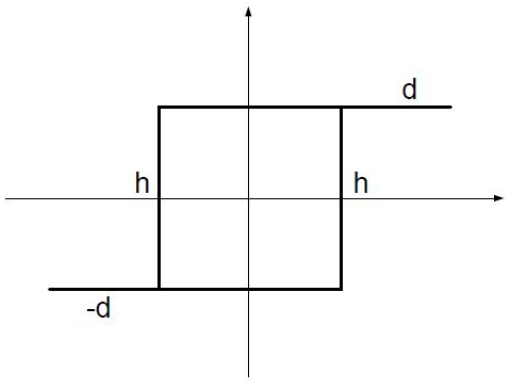



**Figura 7.2.3.2 –** Características del relé en el método de Relay - Feedback.

Tras realizar el método del *relay - feedback* se obtiene una señal oscilante periódica. De esta señal se hallan los parámetros $T_c$ y $a$, como se muestra en la siguiente imagen:

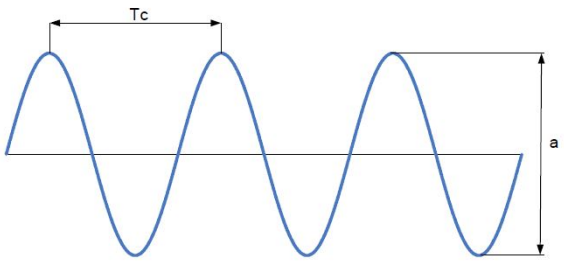



**Figura 7.2.3.3 –** Señal obtenida con el método de Relay - Feedback.

Para obtener el parámetro $K_c$ se hace uso de la siguiente ecuación, en la cual aparece el término $a$ recién calculado [2].

$$K_c = \frac{4 * d}{\pi * \sqrt{a^2 - h^2}} \tag{7.2.3.1}$$

### 7.2.4. Obtención de los parámetros $K_p$, $T_i$ y $T_d$ mediante la aplicación de fórmulas

Los parámetros característicos de un regulador $K_p$, $T_i$ y $T_d$ se obtienen a partir de la ganancia crítica del sistema $(K_c)$ y del periodo de oscilación sostenida $(T_c)$ mediante la resolución de fórmulas matemáticas. Los métodos utilizados en el presente proyecto son las *Fórmulas de Ziegler - Nichols en cadena cerrada* y las *Fórmulas de Tyreus - Luyben*.

#### 7.2.4.1. Ziegler - Nichols [2]

A continuación se muestran las ecuaciones de la **primera aproximación del método de Ziegler - Nichols.**

| Parámetro | Ecuación |
|---|---|
| $K$ | $0{,}6xK_c$ |
| $T_i$ | $0{,}5xT_c$ |
| $T_d$ | $0{,}125xT_c$ |

**Tabla 7.2.4.1 –** Primera aproximación Ziegler - Nichols

La aplicación de estas fórmulas se aprueba como válida cuando se cumple la siguiente relación:

$$2 < kxK_c < 20 \tag{7.2.4.1}$$

La primera aproximación del método de Zieqgler - Nichols es un buen punto de partida para diseñar un regulador, pero en la mayoría de los casos la respuesta del sistema obtenida por este método es mejorable. Con el objetivo de mejorar la respuesta del sistema nacen las fórmulas modificadas de Ziegler Nichols. Estas modificaciones intentan reducir la sobreoscilación obtenida con el método original.

Las **fórmulas de Ziegler - Nichols con poca sobreoscilación** se muestran en la siguiente tabla:

| Parámetro | Ecuación |
|---|---|
| $K$ | $0{,}33xK_c$ |
| $T_i$ | $T_c/2$ |
| $T_d$ | $T_c/3$ |

**Tabla 7.2.4.2 –** Fórmulas Ziegler - Nichols con poca sobreoscilación

Las **fórmulas de Ziegler - Nichols sin sobreoscilación** se muestran a continuación.

| Parámetro | Ecuación |
|---|---|
| $K$ | $0{,}2xK_c$ |
| $T_i$ | $T_c$ |
| $T_d$ | $T_c/3$ |

**Tabla 7.2.4.3 –** Fórmulas Ziegler - Nichols sin sobreoscilación

#### 7.2.4.2. Tyreus - Luyben [2]

A continuación se muestran las ecuaciones del método de Tyreus - Luyben.

| Parámetro | Ecuación |
|---|---|
| $K$ | $0{,}45xK_c$ |
| $T_i$ | 2.2x$T_c$ |
| $T_d$ | $T_c/6{,}3$ |

**Tabla 7.2.4.4 –** Fórmulas Tyreus - Lubin

# 8 Resultados finales

## 8.1. Comunicación entre Easy Java Simulations y Matlab

El primer paso para lograr el control remoto de la planta de nivel de laboratorio es realizar la comunicación entre Easy Java Simulations y Matlab.

Esta comunicación se realiza a través de un servidor creado por la Universidad Nacional de Educación a Distancia (UNED). Este servidor se denomina *RpcMatlabServer* y traduce las órdenes enviadas desde la aplicación de control a Matlab. Las órdenes son enviadas por protocolo *http*.

Las órdenes que pueden ser enviadas a través de este servidor son:

- **matlab.connect()**: Con esta orden se inicia una sesión de Matlab en el ordenador local.
- **matlab.set(”nombre variable”,”valor inicial”)**: Esta orden permite declarar e inicializar variables en la sesión abierta de Matlab.
- **matlab.eval(comandos)**: Permite enviar comandos para que sean ejecutados en la sesión de Matlab iniciada.
- **matlab.get(”variable de Matlab”)**: Permite recoger el valor de una variable utilizada en la sesión activa de Matlab.
- **matlab.disconnect()**: Esta orden finaliza la sesión activa de Matlab.

La carpeta contenedora con el servidor puede estar guardada en cualquier directorio, ya que se accede a ella mediante la ventana de comandos de Windows. En este caso, el servidor se encuentra guardado en el directorio C:Ės de gran importancia incluir el tipo de comunicación y el puerto empleados en la ejecución del ejecutable *RpcMatlabServer.jar*. La instrucción que inicia el servidor es la siguiente:

$$\boxed{java - jar RpcMatlabServer.jar - thttp - p2055}$$

Para lograr una conexión correcta, se debe de tener en cuenta que la dirección IP del ordenador local debe de ser pública y que el puerto utilizado (en este caso 2055) debe de estar abierto y libre.

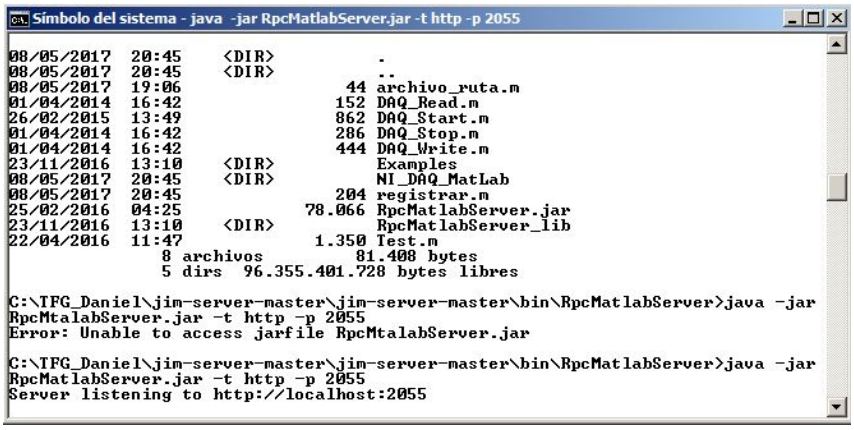



**Figura 8.1.0.1 –** Iniciar servidor a través de comandos de Windows.

Se desarrolla un *script* en un archivo de texto que permite iniciar el servidor sin necesidad de escribir la orden comentada anteriormente. Se introduce el siguiente código en el archivo de texto y se guarda con una extensión *.bat*.

$$\boxed{ECHO java - jar RpcMatlabServer.jar - thttp - p2055}$$

Una vez, iniciado el servidor se debe de configurar el software de desarrollo Easy Java Simulations. Se inicia el programa y se accede a la pantalla de Elementos, situada dentro de la pestaña Modelo. Se debe de abrir la ventana de la figura 7.1.2.8. Se selecciona *SoftwareLinks* y se añade el elemento Matlab.

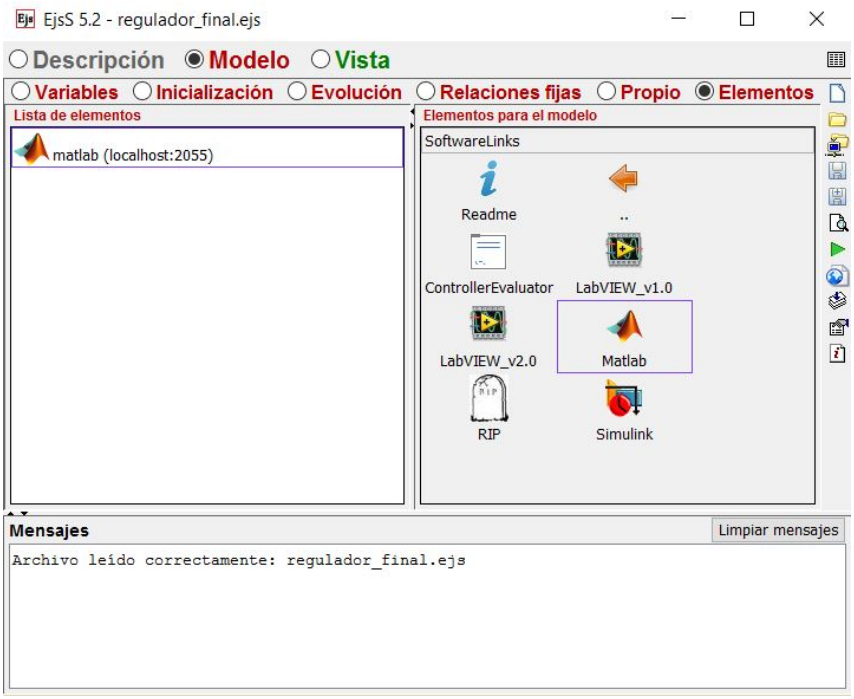



**Figura 8.1.0.2 –** Introducir elemento Matlab.

Para configurar el elemento Matlab se debe de realizar doble click sobre él. Tras realizar esta acción se muestra una ventana de configuración. En ella, aparece por defecto *localhost*,

puerto 2055 y comunicación *http*. En lugar de *localhost* se debe de introducir la dirección IP del ordenador local que tiene instalado el software Matlab, en este caso $193{,}147{,}32{,}125$. El puerto que se utiliza para este proyecto es el configurado por defecto y se debe de cambiar el modo de comunicación a *http*.

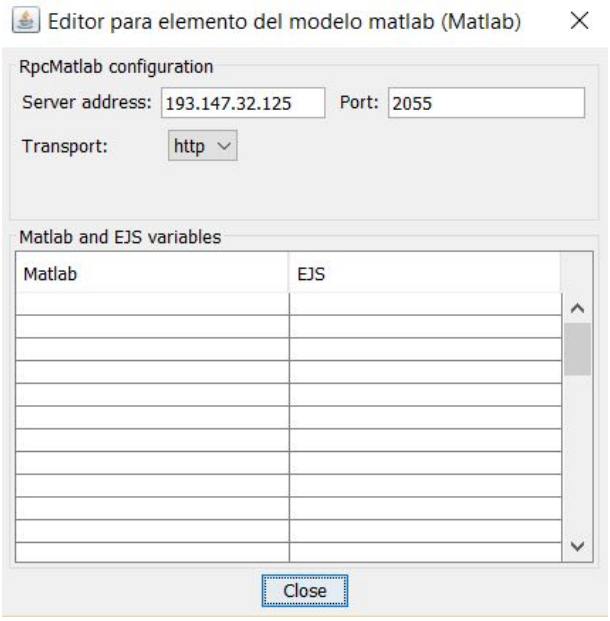



**Figura 8.1.0.3 –** Propiedades del elemento Matlab.

## 8.2. Declaración de variables en EJS

Las variables utilizadas en el proyecto se declaran en el apartado de variables en la pestaña de modelo. Cuando se accede a esta sección se muestra la pantalla de la figura 7.1.2.3.

Para el desarrollo de la aplicación de control de la planta de nivel se hace uso de las siguientes variables:

| Nombre | Valor inicial | Tipo | Descripción |
|---|---|---|---|
| $sp$ | 50 | double | Set point |
| $k$ | 0 | double | Constante proporcional |
| $T_d$ | 0 | double | Constante derivativa |
| $T_i$ | 0 | double | Constante integral |
| $cont\_aux$ | cont_aux | string | Contador de Matlab |
| $cont$ | 0 | double | Contador de Matlab |
| $reguladorP$ | 0 | double | Bandera para regulador P |
| $reguladorPI$ | 0 | double | Bandera para regulador PI |
| $reguladorPD$ | 0 | double | Bandera para regulador PD |
| $reguladorPID$ | 0 | double | Bandera para regulador PID |
| $reguladorPN$ | 0 | double | Bandera para regulador P con N |
| $reguladorPIN$ | 0 | double | Bandera para regulador PI con N |
| $reguladorPDN$ | 0 | double | Bandera para regulador PD con N |
| $reguladorPIDN$ | 0 | double | Bandera para regulador PID con N |
| $reguladorSinN$ | 0 | double | Bandera para reguladores sin N |
| $reguladorConN$ | 0 | double | Bandera para reguladores con N |
| $parametros\_visible$ | false | boolean | Bandera visibilidad parámetros |
| $graficas\_visible$ | false | boolean | Bandera para gráficas |
| $nivel\_ejs$ | 0 | double | Nivel en EJS |
| $k\_string$ | k | string | Nombre constante proporcional Matlab |

| Nombre | Valor inicial | Tipo | Descripción |
|---|---|---|---|
| $sp\_string$ | sp | string | Nombre set point Matlab |
| $T_i\_string$ | Ti | string | Nombre constante integral Matlab |
| $T_d\_string$ | Td | string | Nombre constante derivativa Matlab |
| $limpiar\_grafica$ | false | boolean | Bandera borrar gráfica |
| $min$ | 0 | double | Límite inferior eje X de la gráfica |
| $max$ | 350 | double | Límite superior eje X de la gráfica |
| $autoescale\_x$ | false | boolean | Bandera autoescale de la gráfica |
| $nivel\_visible$ | true | boolean | Bandera visibilidad graficar nivel |
| $sp\_visible$ | false | boolean | Bandera visibilidad graficar set point |
| $senal\_control\_visible$ | false | boolean | Bandera visibilidad graficar señal de control |
| $error\_visible$ | false | boolean | Bandera visibilidad graficar error |
| $C1$ | 0 | double | Parámetro C1 |
| $C2$ | 0 | double | Parámetro C2 |
| $C3$ | 0 | double | Parámetro C3 |
| $C1\_string$ | C1 | string | Nombre parámetro C1 Matlab |
| $C2\_string$ | C2 | string | Nombre parámetro C2 Matlab |
| $C3\_string$ | C3 | string | Nombre parámetro C3 Matlab |
| $N$ | 0 | double | Parámetro N |
| $N\_string$ | N | string | Nombre parámetro N Matlab |
| $error\_ejs$ | 0 | double | Error en EJS |
| $senal\_control\_ejs$ | 0 | double | Señal de control en EJS |
| $error\_ejs$ | 0 | double | Error en EJS |
| $relay\_feedback$ | false | boolean | Bandera método relay feedback |
| $ancho\_ventana$ | 0 | double | Parámetro ancho de ventana |

**Tabla 8.2.0.1 –** Variables de la aplicación de control

El campo correspondiente a dimensión se deja en blanco, ya que en este proyecto no se hace uso de *arrays*.

## 8.3. Programación del cuerpo de la aplicación

El cuerpo del programa se desarrolla en el apartado de evolución, perteneciente a la sección modelo. Esta pantalla se muestra en la figura 7.1.2.5.

En este caso, se realiza la programación en una página de código, por lo que se desarrolla en lenguaje Java.

Se configura esta página para que se ejecute cíclicamente y con un periodo de un segundo. Para realizar esta configuración se debe de introducir un 1 en imágenes por segundo (IPS) y en pasos por visualización (PPV). Se desmarca la opción de arranque para que la evolución no se ejecute al abrir la aplicación, si no que comience mediante un comando.

Con el fin de optimizar la potencia de cálculo de matlab se utilizan matrices fijas de una dimensión de 350 elementos. Estos elementos se guardan en un fichero con extensión *.csv* cada segundo. Para graficar se hace uso del comando *line* de Matlab. Este comando utiliza menos recursos que comandos como el *plot*.

Para enviar las órdenes del regulador al ordenador local se hace uso del comando *matlab.eval()*. Con el fin de optimizar la comunicación con el ordenador local, las órdenes de los reguladores y la representación gráfica se envían en un solo comando *matlab.eval()*. Estas órdenes se deben de separar con punto y coma. Realizando esta acción, se reduce el tiempo de comunicación, ya que únicamente se envía una línea de texto.

Se debe de tener en cuenta que la aplicación de control está diseñada para su uso en el entorno de docencia, por lo que se realizan gráficas tanto en el ordenador local como en el ordenador remoto y a su vez se guardan los datos de las simulaciones en ambos equipos.

La aplicación de control ofrece la posibilidad de regular mediante reguladores sin el parámetro N y con reguladores con el parámetro N. También es posible la realización del método de *Relay - Feedback* para la obtención de los parámetros característicos del regulador.

### 8.3.1. Código de la implementación y representación de los reguladores

Inicialmente se incrementan los contadores de tiempo de Matlab y de EJS, se inicializa la cuenta del periodo de muestre con el comando *tic*, se guarda el tiempo de la simulación en la primera fila y el valor del nivel en la segunda fila de la variable *lectura* y se calculan los errores en el instante inicial.

**Código 8.1:** Inicialización de los reguladores

```
cont=cont+1;
limpiar_grafica=false;
matlab.eval("cont_aux=cont_aux+1;");
```

```
matlab.eval("tic;");
matlab.eval("lectura(1,cont_aux)=cont_aux;");
matlab.eval("lectura(2,cont_aux)=DAQ_Read();")matlab.eval("nivel_matlab=DAQ_Read;");
matlab.eval("error(1,cont_aux)=sp-lectura(2,cont_aux);");
matlab.eval("error_matlab=error(1,cont_aux);";
```

#### 8.3.1.1. Código del instante inicial de los reguladores sin corrección

Para la implantación de los reguladores son necesarios datos en el instante anterior al instante inicial. Este dato se obtiene en la primera ejecución del programa. Tras calcular estos valores, se procede a su representación haciendo uso del comando *line*.

**Código 8.2:** Primera iteracción de los reguladores sin correción

```
if(reguladorSinN==true){
  if(cont==1){
    matlab.eval("u(1,cont_aux)=0;");
    matlab.eval("senal_control_matlab=u(1,cont_aux);");
    matlab.eval("integral(1,cont_aux)=error(1,cont_aux)/2;");
    matlab.eval("DAQ_Write(u(1,cont_aux),0)");
    //Representación
    matlab.eval("figure(1)");
    matlab.eval("line([0 lectura(1,cont_aux)],[lectura(0 lectura(2,cont_aux)]");
    matlab.eval("hold on");
    matlab.eval("line([lectura(0 lectura(1,cont_aux)],[sp sp])");
    matlab.eval("xlim([0 350]");
    matlab.eval("ylim([0 100])");
    matlab.eval("pause(1-toc)";);
    matlab.eval("toc"););
```

#### 8.3.1.2. Código del regulador P sin corrección

Basándose en la ecuación 7.2.2.10, se implementa el regulador P sin corrección.

**Código 8.3:** Regulador P sin correción

```
  if(cont>1){
    Regulador P
    if(reguladorP==1){
      matlab.eval("u(1,cont_aux)=k*error(1,cont_aux);");
      matlab.eval("senal_control_matlab=u(1,cont_aux);");
      matlab.eval("DAQ_Write(u(1,cont_aux),0)");
      //Representación
      matlab.eval("figure(1)");
```

```
      matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
      [lectura(2,cont_aux-1) lectura(2,cont_aux)]");
      matlab.eval("hold on");
      matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp])");
      matlab.eval("xlim([0 350]");
      matlab.eval("ylim([0 100])");
      matlab.eval("pause(1-toc);");
      matlab.eval("toc");
```

#### 8.3.1.3. Código del regulador PI sin corrección

El regulador PI sin corrección rige su funcionamiento por la ecuación 7.2.2.11.

**Código 8.4:** Regulador PI sin correción

```
//Regulador PI
    if(reguladorPI==1){
      matlab.eval("tic");
      matlab.eval("integral(1,cont_aux)=integral(1,cont_aux-1)+(error(1,cont_aux-1)+
      error(1,cont_aux))/2;");
      matlab.eval("u(1,cont_aux)=k*error(1,cont_aux)+k*(1/Ti)*integral(1,cont_aux)
      ;");
      matlab.eval("senal_control_matlab=u(1,cont_aux);");
      matlab.eval("DAQ_Write(u(1,cont_aux),0)");
      //Representación
      matlab.eval("figure(1)");
      matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
      [lectura(2,cont_aux-1) lectura(2,cont_aux)]);");
      matlab.eval("hold on");
      matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp])");
      matlab.eval("ylim([0 100])");
      matlab.eval("xlim([0 350]");
      matlab.eval("pause(1-toc)");
      matlab.eval("toc");
```

#### 8.3.1.4. Código del regulador PD sin corrección

Haciendo uso de la ecuación 7.2.2.13, se implementa el regulador PD sin corrección.

**Código 8.5:** Regulador PD sin correción

```
//Regulador PD
    if(reguladorPD==1){
      matlab.eval("u(1,cont_aux)=k*error(1,cont_aux)+k*(Td/1)*(error(1,cont_aux)-
```

```
      error(1,cont_aux-1));");
      matlab.eval("senal_control_matlab=u(1,cont_aux);");
      matlab.eval("DAQ_Write(u(1,cont_aux),0)");
      //Representación
      matlab.eval("figure(1)");
      matlab.eval("plot(lectura(1,:),lectura(2,:),'-r')");
      matlab.eval("hold on");
      matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
      [lectura(2,cont_aux-1) lectura(2,cont_aux)]);");
      matlab.eval("hold on");
      matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp])");
      matlab.eval("ylim([0 100])");
      matlab.eval("xlim([0 350]");
      matlab.eval("pause(1-toc)");
      matlab.eval("toc");
```

#### 8.3.1.5. Código del regulador PID sin corrección

El funcionamiento del regulador PID sin corrección está descrito por la ecuación 7.2.2.2.

**Código 8.6:** Regulador PID sin correción

```
//Regulador PID
    if(reguladorPID==1){
      matlab.eval("tic");
      matlab.eval("integral(1,cont_aux)=integral(1,cont_aux-1)+(error(1,cont_aux-1)+
      error(1,cont_aux))/2;");
      matlab.eval("u(1,cont_aux)=k*error(1,cont_aux)+k*(Td/1)*(error(1,cont_aux)-
      error(1,cont_aux-1))+k*(1/Ti)*integral(1,cont_aux);");
      matlab.eval("senal_control_matlab=u(1,cont_aux);");
      matlab.eval("DAQ_Write(u(1,cont_aux),0)");
      //Representación
      matlab.eval("figure(1)");
      matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
      [lectura(2,cont_aux-1) lectura(2,cont_aux)]);");
      matlab.eval("hold on");
      matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp])");
      matlab.eval("ylim([0 100])");
      matlab.eval("xlim([0 350]");
      matlab.eval("pause(1-toc)");
      matlab.eval("toc");
```

#### 8.3.1.6. Código del instante inicial de los reguladores con corrección

Para la implantación de los reguladores son necesarios datos en el instante anterior al instante inicial. Este dato se obtiene en la primera ejecución del programa. Tras calcular estos

valores, se procede a su representación haciendo uso del comando *line*.

**Código 8.7:** Primera iteración reguladores con correción

```
if (cont==1){
matlab.eval("u(1,cont_aux)=0;");
matlab.eval("senal_control_matlab=u(1,cont_aux);");
matlab.eval("integral(1,cont_aux)=error(1,cont_aux)*C1;");
matlab.eval("derivada(1,cont_aux)=error(1,cont_aux)*C2;");
matlab.eval("DAQ_Write(u(1,cont_aux),0)");
//Representación
matlab.eval("figure(1)");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);");
matlab.eval("hold on");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp])");
matlab.eval("ylim([0 100])");
matlab.eval("xlim([0 350]");
matlab.eval("pause(1-toc)");
matlab.eval("toc");
```

#### 8.3.1.7. Código del regulador P con corrección

La ecuación que rige el funcionamiento del regulador P con corrección es idéntica a la del regulador P sin corrección, por lo tanto la implementación es común para ambos casos.

#### 8.3.1.8. Código del regulador PI con corrección

El funcionamiento del regulador PI con corrección está definido por la ecuación 7.2.2.12.

**Código 8.8:** Regulador PI con correción

```
matlab.eval("integral(1,cont_aux)=C1*(error(1,cont_aux)+error(1,cont_aux-1))+
integral(1,cont_aux-1);");
matlab.eval("u(1,cont_aux)=k*(error(1,cont_aux)+integral(1,cont_aux));");
matlab.eval("senal_control_matlab=u(1,cont_aux);");
matlab.eval("DAQ_Write(u(1,cont_aux),0)");
//Representación
matlab.eval("figure(1)");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);");
matlab.eval("hold on");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp])");
matlab.eval("ylim([0 100])");
matlab.eval("xlim([0 350]");
```

```
matlab.eval("pause(1-toc)");
matlab.eval("toc");
```

#### 8.3.1.9. Código del regulador PD con corrección

La implementación del regulador PD con corrección se realiza a partir de la ecuación 7.2.2.14.

**Código 8.9:** Regulador PD con correción

```
matlab.eval("derivada(1,cont_aux)=C2*(error(1,cont_aux)+error(1,cont_aux-1))+
C3*derivada(1,cont_aux-1);");
matlab.eval("u(1,cont_aux)=k*(error(1,cont_aux)+derivada(1,cont_aux));");
matlab.eval("senal_control_matlab=u(1,cont_aux);");
matlab.eval("DAQ_Write(u(1,cont_aux),0)");
//Representación
matlab.eval("figure(1)");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);");
matlab.eval("hold on");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp])");
matlab.eval("ylim([0 100])");
matlab.eval("xlim([0 350]");
matlab.eval("pause(1-toc)");
matlab.eval("toc");
```

#### 8.3.1.10. Código del regulador PID con corrección

La ecuación 7.2.2.6 rige el comportamiento del regulador PID con corrección.

**Código 8.10:** Regulador PID con correción

```
if (reguladorPIDN==1){
matlab.eval("integral(1,cont_aux)=C1*(error(1,cont_aux)+error(1,cont_aux-1))+
integral(1,cont_aux-1);");
matlab.eval("derivada(1,cont_aux)=C2*(error(1,cont_aux)+error(1,cont_aux-1))+
C3*derivada(1,cont_aux-1);");
matlab.eval("u(1,cont_aux)=k*(error(1,cont_aux)+
integral(1,cont_aux)+derivada(1,cont_aux));");
matlab.eval("senal_control_matlab=u(1,cont_aux);");
matlab.eval("DAQ_Write(u(1,cont_aux),0)");
//Representación
matlab.eval("figure(1)");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
```

```
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);");
matlab.eval("hold on");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp])");
matlab.eval("ylim([0 100])");
matlab.eval("xlim([0 350]");
matlab.eval("pause(1-toc)");
matlab.eval("toc");
```

### 8.3.2. Código de la implementación del método de Relay - Feedback

Para realizar el método del Relay - Feedback es necesario someter a la planta al efecto de un relé. Esto se muestra en la figura 7.2.3.1. El código que permite emular el efecto del relé se muestra a continuación:

**Código 8.11:** Método Relay - Feedback

```
if(relay_feedback==true){
if(nivel_ejs < (sp-ancho_ventana)){
matlab.eval("u(1,cont_aux)=100;");
matlab.eval("senal_control_matlab=u(1,cont_aux);");
matlab.eval("DAQ_Write(u(1,cont_aux),0)");
//Representación
matlab.eval("figure(1)");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);");
matlab.eval("pause(1-toc)");
matlab.eval("toc");


else if(nivel_ejs > (sp+ancho_ventana)){
matlab.eval("u(1,cont_aux)=0;");
matlab.eval("senal_control_matlab=u(1,cont_aux);");
matlab.eval("DAQ_Write(u(1,cont_aux),0);");
//Representación
matlab.eval("figure(1)");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);");
matlab.eval("pause(1-toc)");
matlab.eval("toc");

else{
matlab.eval("figure(1);");
matlab.eval("line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);");
matlab.eval("pause(1-toc)");
matlab.eval("toc");
}
```

### 8.3.3. Adquisición de variables de Matlab

Es necesario adquirir el valor de las variables *nivel, error* y *señal de control* para graficar su valor en el Easy Java Simulations. Estos valores también se almacenan en un archivo *.csv*. La instrucción de la que se hace uso es *matlab.get()*.

#### 8.3.3.1. Código adquisición de la variable *nivel*

**Código 8.12:** Adquisición variable nivel

```
String[] varsToGet = new String[]{"nivel_matlab"};
Object[] result = matlab.get(varsToGet);
Double[] y_array = (Double[])result[0];
nivel_ejs = y_array[0].doubleValue();
```

#### 8.3.3.2. Código adquisición de la variable *error*

**Código 8.13:** Adquisición variable error

```
varsToGet = new String[]{"error_matlab"};
result = matlab.get(varsToGet);
y_array = (Double[])result[0];
error_ejs = y_array[0].doubleValue();
```

#### 8.3.3.3. Código adquisición de la variable *señal de control*

**Código 8.14:** Adquisición varianble señal de control

```
varsToGet = new String[]{"senal_control_matlab"};
result = matlab.get(varsToGet);
y_array = (Double[])result[0];
senal_control_ejs = y_array[0].doubleValue();
```

### 8.3.4. Almacenamiento de datos

Se crea una base de datos en el ordenador local y una base de datos en el ordenador remoto. Las variables que se almacenan son el tiempo, el nivel del depósito, el error y la señal de control. El formato de estas bases de datos es *.csv*.

#### 8.3.4.1. Código de la base de datos del ordenador remoto

**Código 8.15:** Base de datos en el ordenador remoto

```
//Guardar datos en fichero csv ordenador remoto
if(cont==1){
_saveText("base_datos.csv","Tiempo;Set Point;Nivel;Error;Senal de control\n"+cont+";
"+sp+";"+nivel_ejs+";"+error_ejs+";"+senal_control_ejs);
}else{
memoria=_readText("base_datos.csv");
_saveText("base_datos.csv",memoria+cont+";"+sp+";"+nivel_ejs+";"+error_ejs+";
"+senal_control_ejs+"\n");
}
```

#### 8.3.4.2. Código de la base de datos del ordenador local

Para realizar la base de datos en el ordenador local, se crea una función en Matlab en el propio ordenador.

**Código 8.16:** Base de datos en el ordenador local

```
function registrar(fileID,cont_aux,sp,nivel_matlab,error_matlab,
senal_control_matlab)
fprintf(fileID,'\n%f %f %f %f %f',[cont_aux; sp; nivel_matlab;
error_matlab; senal_control_matlab;]);
end
```

Esta función es llamada con un periodo de 1 segundo y se le aportan los datos a almacenar.

**Código 8.17:** Función de matlab de almacenamiento de datos

```
//Guardar datos en fichero csv en ordenador local
if(cont==1){
//matlab.eval("fileID=fopen(ruta,'w')");
//matlab.eval("fprintf(fileID,'%s %s %s %s %s',['Tiempo;' 'Set Point;' 'Nivel;'
 'Error;' 'Senal de Control;']);");
//matlab.eval("registrar(fileID,cont_aux,sp,nivel_matlab,error_matlab,
senal_control_matlab);");
}else{
matlab.eval("registrar(fileID,cont_aux,sp,nivel_matlab,error_matlab,
senal_control_matlab);");
}
```

### 8.3.5. Código completo de la página Evolución

**Código 8.18:** Desarrollo completo de la página Evolución

```
cont=cont+1;
limpiar_grafica=false;
matlab.eval("cont_aux=cont_aux+1;tic;lectura(1,cont_aux)=cont_aux;
lectura(2,cont_aux)=DAQ_Read();nivel_matlab=DAQ_Read;
error(1,cont_aux)=sp-lectura(2,cont_aux);error_matlab=error(1,cont_aux);");

if(reguladorSinN==true){
if(cont==1){
matlab.eval("u(1,cont_aux)=0;senal_control_matlab=u(1,cont_aux);
integral(1,cont_aux)=error(1,cont_aux)/2;DAQ_Write(u(1,cont_aux),0);figure(1);
line([0 lectura(1,cont_aux)],[0 lectura(2,cont_aux)]);xlim([0 350]);ylim([0 100]);
pause(1-toc);toc;");
}

if(cont>1){
if(reguladorP==1){
matlab.eval("u(1,cont_aux)=k*error(1,cont_aux);senal_control_matlab=u(1,cont_aux);
DAQ_Write(u(1,cont_aux),0);figure(1);
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);hold on;
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp]);xlim([0 350]);
ylim([0 100]);
pause(1-toc);toc;");
}
//Regulador PD
if(reguladorPD==1){
matlab.eval("u(1,cont_aux)=k*error(1,cont_aux)+k*(Td/1)*(error(1,cont_aux)-
error(1,cont_aux-1));senal_control_matlab=u(1,cont_aux);DAQ_Write(u(1,cont_aux),0);
figure(1);line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);hold on;
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[sp sp]);xlim([0 350]);ylim([0 100]);pause(1-toc);toc;");
}

//Regulador PI
if(reguladorPI==1){
matlab.eval("integral(1,cont_aux)=integral(1,cont_aux-1)+(error(1,cont_aux-1)+
error(1,cont_aux))/2;
u(1,cont_aux)=k*error(1,cont_aux)+k*(1/Ti)*integral(1,cont_aux);
senal_control_matlab=u(1,cont_aux);
DAQ_Write(u(1,cont_aux),0);figure(1);
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);hold on;
```

```
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp]);xlim([0 350]);
ylim([0 100]);
pause(1-toc);toc;");
}
//Regulador PID
if(reguladorPID==1){
matlab.eval("integral(1,cont_aux)=integral(1,cont_aux-1)+
(error(1,cont_aux-1)+error(1,cont_aux))/2;u(1,cont_aux)=
k*error(1,cont_aux)+k*(Td/1)*(error(1,cont_aux)-
error(1,cont_aux-1))+k*(1/Ti)*integral(1,cont_aux);
senal_control_matlab=u(1,cont_aux);
DAQ_Write(u(1,cont_aux),0);figure(1);
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);hold on;
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp]);
xlim([0 350]);ylim([0 100]);
pause(1-toc);toc;");
}
}
}

if (reguladorConN==true){
if (cont==1){
matlab.eval("u(1,cont_aux)=0;senal_control_matlab=u(1,cont_aux);
integral(1,cont_aux)=error(1,cont_aux)*C1;
derivada(1,cont_aux)=error(1,cont_aux)*C2;DAQ_Write(u(1,cont_aux),0);
figure(1);line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);hold on;
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp]);
xlim([0 350]);ylim([0 100]);
pause(1-toc);toc;");
}

if (cont>1){
if (reguladorPN==1){
matlab.eval("u(1,cont_aux)=k*(error(1,cont_aux));senal_control_matlab=
u(1,cont_aux);
DAQ_Write(u(1,cont_aux),0);figure(1);
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);hold on;
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp]);xlim([0 350]);
ylim([0 100]);pause(1-toc);toc;");
}
if (reguladorPDN==1){
matlab.eval("derivada(1,cont_aux)=C2*(error(1,cont_aux)+error(1,cont_aux-1))+
C3*derivada(1,cont_aux-1);u(1,cont_aux)=k*(error(1,cont_aux)+derivada(1,cont_aux));
senal_control_matlab=u(1,cont_aux);DAQ_Write(u(1,cont_aux),0);figure(1);
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);hold on;
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp]);xlim([0 350]);
```

```
ylim([0 100]);
pause(1-toc);toc;");
}
if (reguladorPIN==1){
matlab.eval("integral(1,cont_aux)=C1*(error(1,cont_aux)+error(1,cont_aux-1))+
integral(1,cont_aux-1);u(1,cont_aux)=k*(error(1,cont_aux)+integral(1,cont_aux));
senal_control_matlab=u(1,cont_aux);DAQ_Write(u(1,cont_aux),0);figure(1);
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);hold on;
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp]);xlim([0 350]);
ylim([0 100]);
pause(1-toc);toc;");
}
if (reguladorPIDN==1){
matlab.eval("integral(1,cont_aux)=C1*(error(1,cont_aux)+error(1,cont_aux-1))+
integral(1,cont_aux-1);derivada(1,cont_aux)=C2*(error(1,cont_aux)+
error(1,cont_aux-1))+C3*derivada(1,cont_aux-1);u(1,cont_aux)=k*(error(1,cont_aux)+
integral(1,cont_aux)+derivada(1,cont_aux));senal_control_matlab=u(1,cont_aux);
DAQ_Write(u(1,cont_aux),0);figure(1);
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);hold on;
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],[sp sp]);xlim([0 350]);
ylim([0 100]);
pause(1-toc);toc;");
}
}
}

if(relay_feedback==true){
if(nivel_ejs < (sp-ancho_ventana)){
matlab.eval("u(1,cont_aux)=100;senal_control_matlab=u(1,cont_aux);
DAQ_Write(u(1,cont_aux),0);
figure(1);line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);xlim([0 350]);ylim([0 100]);
pause(1-toc);toc;");
}

else if(nivel_ejs > (sp+ancho_ventana)){
matlab.eval("u(1,cont_aux)=0;senal_control_matlab=u(1,cont_aux);
DAQ_Write(u(1,cont_aux),0);figure(1);
line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);xlim([0 350]);
ylim([0 100]);pause(1-toc);toc;");
}
else{matlab.eval("figure(1);line([lectura(1,cont_aux-1) lectura(1,cont_aux)],
[lectura(2,cont_aux-1) lectura(2,cont_aux)]);xlim([0 350]);ylim([0 100]);
pause(1-toc);toc;");}
}
//Recogida de datos
String[] varsToGet = new String[]{"nivel_matlab"};
```

```
Object[] result = matlab.get(varsToGet);
Double[] y_array = (Double[])result[0];
nivel_ejs = y_array[0].doubleValue();

varsToGet = new String[]{"error_matlab"};
result = matlab.get(varsToGet);
y_array = (Double[])result[0];
error_ejs = y_array[0].doubleValue();

varsToGet = new String[]{"senal_control_matlab"};
result = matlab.get(varsToGet);
y_array = (Double[])result[0];
senal_control_ejs = y_array[0].doubleValue();

//Guardar datos en fichero csv ordenador remoto
if(cont==1){
_saveText("base_datos.csv","Tiempo;Set Point;Nivel;Error;
Senal de control\n"+cont+";"+sp+";"+nivel_ejs+";
"+error_ejs+";"+senal_control_ejs);
}else{
memoria=_readText("base_datos.csv");
_saveText("base_datos.csv",memoria+cont+";"+sp+";"+nivel_ejs+";"+error_ejs+";"+
senal_control_ejs+"\n");
}

//Guardar datos en fichero csv en ordenador local
if(cont==1){
matlab.eval("fileID=fopen(ruta,'w');fprintf(fileID,'%s %s %s %s %s',['Tiempo;'
 'Set Point;'
'Nivel;' 'Error;' 'Senal deControl;']);registrar(fileID,cont_aux,sp,nivel_matlab,
error_matlab,senal_control_matlab);");
}else{
matlab.eval("registrar(fileID,cont_aux,sp,nivel_matlab,
error_matlab,senal_control_matlab);");
}
```

## 8.4. *Interface* de la aplicación

La *interface* de la aplicación de control se compone por cinco pantallas: pantalla principal, pantalla de inserción de los parámetros propios del regulador, pantalla de inserción de la consigna y del parámetro corrector N, pantalla de inserción de los parámetros para el método del Relay - Feedback y la pantalla de gráficos.

### 8.4.1. Pantalla principal

La pantalla principal se muestra en la siguiente imagen:

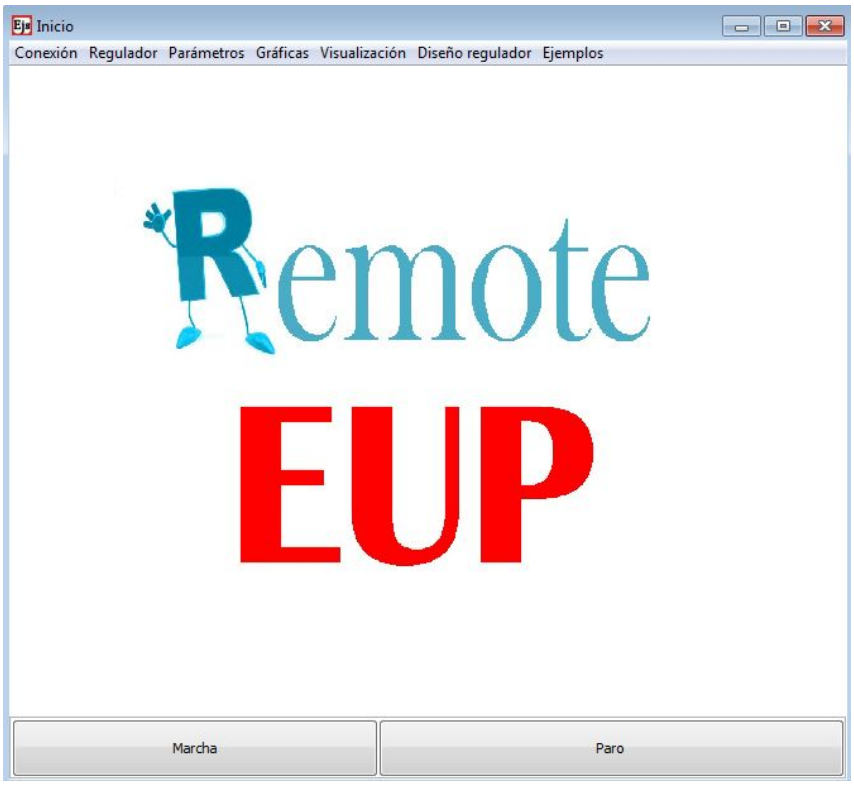



**Figura 8.4.1.1 –** Pantalla principal de la aplicación de control.

La pantalla principal la compone un menú de navegación (Conexión, Regulador, Parámetros, Gráficas, Visualización,Diseño regulador y Ejemplos), el logo de la aplicación de control y los botones de marcha y paro del regulador.

#### 8.4.1.1. Menu: Conexión

El menú conexión es desplegable y permite el inicio y el cierre de una sesión de Matlab en el ordenador local.


**Figura 8.4.1.2 –** Sub-menú Conexión.

En el **Botón Conectar** se programa la conexión con Matlab, se declaran e inicializan las variables y se inicia la comunicación con la tarjeta de adquisición de datos.

**Código 8.19:** Botón Conectar

```
//Conexión con Matlab
matlab.connect();
```

```
//Declaracion en Matlab
matlab.eval("lectura=zeros([2,350]);");
matlab.eval("error=zeros([1,350]);");
matlab.eval("u=zeros([1,350]);");
matlab.eval("integral=zeros([1,350]);");
matlab.eval("derivada=zeros([1,350]);");

matlab.eval("nombre='prueba';");

//Inicialización de variables
matlab.set(cont_aux,0);


//Iniciación de la tarjeta DAQ
matlab.eval("DAQ_Start");
```

En el **Botón Desconectar**, se escribe una señal de control nula sobre el sistema, se detiene la comunicación con la tarjeta de adquisición de datos y se cierra la sesión activa de Matlab.

**Código 8.20:** Botón desconectar

```
//Desconexión
matlab.eval("DAQ_Write(0,0)");
matlab.eval("DAQ_Stop");
matlab.disconnect();
```

#### 8.4.1.2. Menú: Regulador

En el menú Regulador se elije el regulador a implementar. Las opciones que se ofrecen son el regulador PID sin corrección y sus simplificaciones y el regulador PID con corrección y sus simplificaciones.

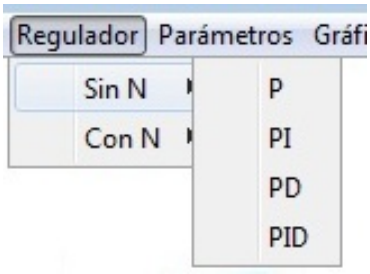



**Figura 8.4.1.3 –** Sub-menú Regulador.

La programación de los botones de elección de los reguladores es similar para todos los casos. Se indica si se ha elegido un regulador con corrección o sin corrección y el tipo de

regulador que se ha escogido.

El código del **Botón del regulador P sin N** es el siguiente:

**Código 8.21:** Botón regulador P sin N

```
reguladorP=1;
reguladorPI=0;
reguladorPD=0;
reguladorPID=0;

reguladorSinN=true;
reguladorConN=false;
relay_feedback=false;
```

El código del **Botón del regulador PI sin N** se muestra a continuación:

**Código 8.22:** Botón regulador PI sin N

```
reguladorP=0;
reguladorPI=1;
reguladorPD=0;
reguladorPID=0;

reguladorSinN=true;
reguladorConN=false;
relay_feedback=false;
```

El **Botón del regulador PD sin N** contiene el siguiente código:

**Código 8.23:** Botón regulador PD sin N

```
reguladorP=0;
reguladorPI=0;
reguladorPD=1;
reguladorPID=0;

reguladorSinN=true;
reguladorConN=false;
relay_feedback=false;
```

El código del **Botón del regulador PID sin N** es el siguiente:

**Código 8.24:** Botón regulador PID sin N

```
reguladorP=0;
reguladorPI=0;
reguladorPD=0;
reguladorPID=1;

reguladorSinN=true;
reguladorConN=false;
relay_feedback=false;
```

El código del **Botón del regulador P con N** es el siguiente:

**Código 8.25:** Botón regulador P con N

```
reguladorPN=1;
reguladorPIN=0;
reguladorPDN=0;
reguladorPIDN=0;

reguladorSinN=false;
reguladorConN=true;
relay_feedback=false;
```

El código del **Botón del regulador PI con N** se muestra a continuación:

**Código 8.26:** Botón regulador PI con N

```
reguladorPN=0;
reguladorPIN=1;
reguladorPDN=0;
reguladorPIDN=0;

reguladorSinN=false;
reguladorConN=true;
relay_feedback=false;
```

El **Botón del regulador PD con N** contiene el siguiente código:

**Código 8.27:** Botón regulador PD con N

```
reguladorPN=0;
reguladorPIN=0;
```

```
reguladorPDN=1;
reguladorPIDN=0;

reguladorSinN=false;
reguladorConN=true;
relay_feedback=false;
```

El código del **Botón del regulador PID con N** es el siguiente:

**Código 8.28:** Botón regulador PID con N

```
reguladorPN=0;
reguladorPIN=0;
reguladorPDN=0;
reguladorPIDN=1;

reguladorSinN=false;
reguladorConN=true;
relay_feedback=false;
```

#### 8.4.1.3. Menú: Parámetros

El menú de Parámetros contiene un sub-menú que permite abrir las ventanas de inserción de parámetros para el regulador seleccionado o para el método del Relay - Feedback.

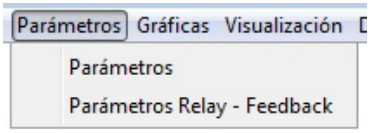



**Figura 8.4.1.4 –** Sub-menú Parámetros.

En el **Botón Parámetros** se activa una variable para abrir la ventana de inserción de parámetros del regulador.

**Código 8.29:** Botón parámetros

```
parametros_visible=true;
parametros_RF_visible=false;
```

En el **Botón Parámetros Relay - Feedback** se activa una variable que rige la apertura y el cierre de la ventana de inserción de parámetros del método de Relay - Feedback.

**Código 8.30:** Botón parámetros Relay - Feedback

```
parametros_RF_visible=true;
parametros_visible=false;
```

#### 8.4.1.4. Menú: Gráficas

Desde el menú de gráficas se permite la apertura de la ventana de gráficas.

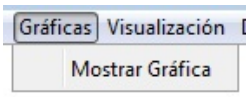



**Figura 8.4.1.5 –** Sub-menú Gráfica.

En el **Botón de Mostrar Gráfica** se activa una variable para permitir la apertura de la ventana de gráficas.

**Código 8.31:** Botón mostrar gráfica

```
graficas_visible=true;
```

#### 8.4.1.5. Menú: Visualización

En el menú de Visualización se permite ver en directo el estado de la planta. Esta visualización se realiza a través del explorador de internet predeterminado del usuario.

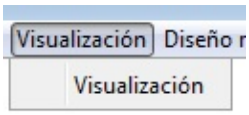



**Figura 8.4.1.6 –** Sub-menú Visualización.

En el **Botón de Visualización** se programa mediante código Java la apertura del explorador predeterminado en la *URL* de transmisión de la WebCam.

**Código 8.32:** Botón Visualización

```
try {
Desktop.getDesktop().browse(new URI("http://193.147.32.125:85/"));
} catch (Exception ex) {
JOptionPane.showMessageDialog(null, "No se ha podido cargar la página");
}
```

#### 8.4.1.6. Menú: Diseño regulador

En el menú de Diseño de regulador se permite seleccionar el método del Relay - Feedback para la obtención de los parámetros propios del regulador.

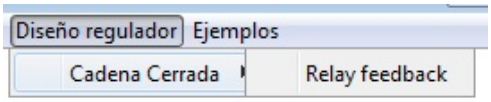



**Figura 8.4.1.7 –** Sub-menú Diseño.

En el **Botón Relay - feedback** se activa una variable que indica la selección del método.

**Código 8.33:** Botón Relay - Feedback

```
relay_feedback=true;
reguladorSinN=false;
reguladorConN=false;
```

#### 8.4.1.7. Menú: Ejemplos

El menú de ejemplos contiene un sub-menú que permite elegir las ecuaciones utilizadas para la obtención de los parámetros propios del regulador. Se puede elegir entre: Ziegler - Nichols, Ziegler - Nichols No Overshoot, Ziegler - Nichols Some Overshoot y Tyreus - Luyben. Los reguladores implementados en los ejemplos son PID sin corrección.


**Figura 8.4.1.8 –** Sub-menú Ejemplos.

El código del **Botón Ziegler - Nichols** es el siguiente:

**Código 8.34:** Botón Ziegler - Nichols

```
reguladorSinN=true;
reguladorPID=1;

sp=50;
k=1.25;
```

```
Ti=15;
Td=3.75;
matlab.set(k_string,k);
matlab.set(Ti_string,Ti);
matlab.set(Td_string,Td);
matlab.set(sp_string,sp);
```

El **Botón Z&N No Overshoot** contiene el siguiente código:

**Código 8.35:** Botón Z&N No Overshoot

```
reguladorSinN=true;
reguladorPID=1;

sp=50;
k=0.42;
Ti=30;
Td=10;
matlab.set(k_string,k);
matlab.set(Ti_string,Ti);
matlab.set(Td_string,Td);
matlab.set(sp_string,sp);
```

El código del **Botón Z&N Some Overshoot** se muestra a continuación:

**Código 8.36:** Botón Z&N Some Overshoot

```
reguladorSinN=true;
reguladorPID=1;

sp=50;
k=0.69;
Ti=15;
Td=10;
matlab.set(k_string,k);
matlab.set(Ti_string,Ti);
matlab.set(Td_string,Td);
matlab.set(sp_string,sp);
```

El **Botón Tyreus - Luyben** contiene el siguiente código:

**Código 8.37:** Botón Tyreus - Luyben

```
reguladorSinN=true;
```

```
reguladorPID=1;

sp=50;
k=0.943;
Ti=66;
Td=4.762;
matlab.set(k_string,k);
matlab.set(Ti_string,Ti);
matlab.set(Td_string,Td);
matlab.set(sp_string,sp);
```

#### 8.4.1.8. Panel de Mando: Marcha

En el botón de marcha se limpia la gráfica, se inicializan los ejes de la misma y se pone en marcha la simulación.

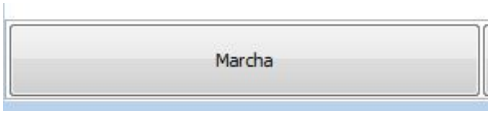



**Figura 8.4.1.9 –** Panel de Mando: Marcha.

El código que contiene el botón de marcha se muestra a continuación:

**Código 8.38:** Botón marcha

```
//Limpiar gráficas
min=0;
max=350;
limpiar_grafica=true;
matlab.eval("cla reset");
matlab.eval("archivo_ruta");
_play()
```

#### 8.4.1.9. Panel de Mando: Paro

En el botón de paro se inicializan todas las variables del proceso y se para la simulación.

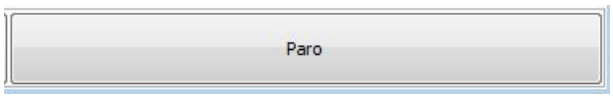



**Figura 8.4.1.10 –** Panel de Mando: Paro.

El botón de paro contiene el siguiente código:

**Código 8.39:** Botón Paro

```
matlab.eval("DAQ_Write(0,0)");
matlab.eval("sp=0;");
matlab.eval("k=0;");
matlab.eval("Ti=0;");
matlab.eval("Td=0;");
matlab.eval("C1=0;");
matlab.eval("C2=0;");
matlab.eval("C3=0;");
matlab.eval("N=0;");
matlab.eval("ancho_ventana=0;");
matlab.eval("lectura=zeros([2,350]);");
matlab.eval("error=zeros([1,350]);");
matlab.eval("u=zeros([1,350]);");
matlab.eval("integral=zeros([1,350]);");
matlab.eval("derivada=zeros([1,350]);");
matlab.eval("nivel_ejs=0;");
matlab.eval("error_ejs=0;");
matlab.eval("senal_control_ejs=0;");
matlab.set(cont_aux,0);
cont=0;
reguladorP=0;
reguladorPI=0;
reguladorPD=0;
reguladorPID=0;
reguladorPN=0;
reguladorPIN=0;
reguladorPDN=0;
reguladorPIDN=0;
relay_feedback=false;
reguladorSinN=false;
reguladorConN=false;
matlab.eval("fclose(fileID);");
_pause()
```

### 8.4.2. Ventana inserción parámetros del Regulador

La pantalla de inserción de los parámetros del regulador se abre al realizar click sobre el botón *Parámetros*, mostrado en la figura 8.4.1.4.

El aspecto de esta pantalla se muestra a continuación:

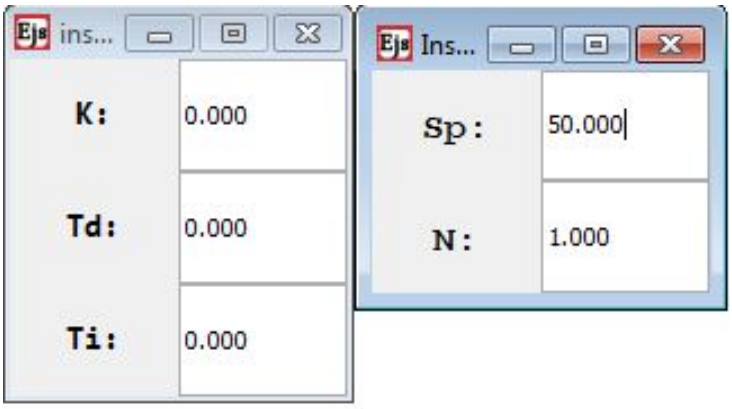



**Figura 8.4.2.1 –** Ventana de inserción de parámetros del Regulador.

Esta ventana se muestra cuando se cumple la condición *variables_visible=true*. Está formada por 5 campos de inserción de valor: k, Td, Ti, Sp y N.

Las propiedades que muestra un campo numérico son las siguientes:

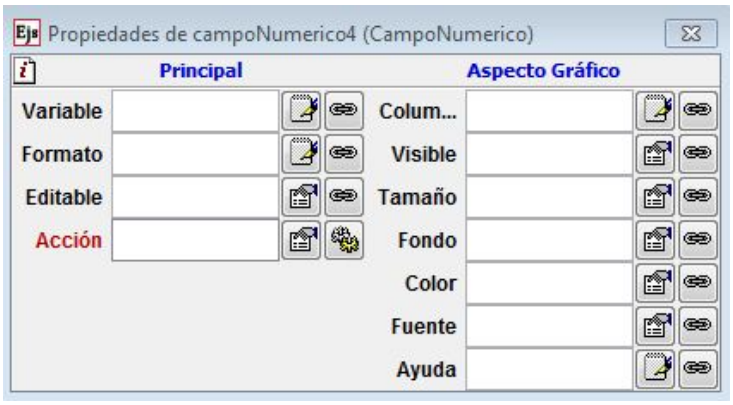



**Figura 8.4.2.2 –** Ventana configuración campo numérico.

### 8.4.2.1. Campo de inserción parámetro K

En el campo de inserción del parámetro K se configuran los elementos *variable* y *Acción*. En el apartado *variable* se inserta la variable k y en el apartado *Acción* se introduce el siguiente código:

**Código 8.40:** Campo de inserción del parámetro K

```
matlab.set(k_string,k);
```

### 8.4.2.2. Campo de inserción parámetro Td

En el campo de inserción del parámetro Td se configura el elemento *Variable* con la variable Td y en el elemento *Acción* se inicializa el valor Td y se calcula el valor de la variable C2 y C3 si el parámetro N es distinto de 0.

**Código 8.41:** Campo de inserción del parámetro Td

```
matlab.set(Td_string,Td);

if(N!=0){
C2=(2*Td)/(2*(Td/N)+1);
C3=((2*Td/N)-1)/((2*Td/N)+1);
matlab.set(C2_string,C2);
matlab.set(C3_string,C3);
}
```

#### 8.4.2.3. Campo de inserción parámetro Ti

En el campo de inserción del parámetro Ti se configura el elemento *Variable* con la variable Ti y en el elemento *Acción* se introduce el siguiente código:

**Código 8.42:** Campo de inserción del parámetro Ti

```
matlab.set(Ti_string,Ti);

if(Ti!=0){
C1=1/(2*Ti);
matlab.set(C1_string,C1);
}
```

#### 8.4.2.4. Campo de inserción Consigna

En el campo de inserción de la consigna se asocia la variable sp al elemento *Variable* y en el elemento *acción* se inicializa esta variable en Matlab.

**Código 8.43:** Campo de inserción de la consigna

```
matlab.set(sp_string,sp);
```

#### 8.4.2.5. Campo de inserción parámetro N

En el campo de inserción del parámetro N se asocia la variable N con el elemento *Variable* y en el elemento *Acción* se inicializa esta variable y las variables C1 y C2 en Matlab.

**Código 8.44:** Campo de inserción del parámetro N

```
matlab.set(N_string,N);

if(C2!=0){
C2=(2*Td)/(2*(Td/N)+1);
matlab.set(C2_string,C2);
}
if(C3!=0){
C3=((2*Td/N)-1)/((2*Td/N)+1);
matlab.set(C3_string,C3);
}
```

### 8.4.3. Ventana de inserción de parámetros de Relay - Feedback

La ventana de inserción de los parámetros de Relay - Feedback se muestra cuando la variable *parametros_RF_visible=true*. En esta ventana se insertan el valor de las variables necesarias para la realización del método de Relay - Feedback.

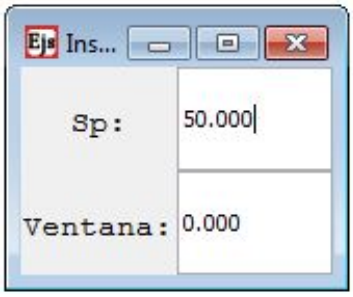



**Figura 8.4.3.1 –** Ventana inserción parámetros Relay - Feedback.

Esta ventana se compone por dos campos numéricos de inserción, uno está destinado a la consigna y el otro al ancho de ventana.

#### 8.4.3.1. Campo de inserción de la consigna

La configuración del campo de inserción de la consigna es similar al explicado con anterioridad en la ventana de inserción de parámetros del regulador.

#### 8.4.3.2. Campo de inserción del ancho de ventana

En el campo de inserción del ancho de ventana se vincula la propiedad *Variable* a la variable ancho_ventana. La propiedad de *Acción* se configura con el siguiente código:

**Código 8.45:** Campo de inserción del ancho de ventana

```
matlab.set(ancho_ventana_string,ancho_ventana);
```

### 8.4.4. Pantalla de gráficos

La pantalla de gráficos está destinada a mostrar la evolución de las señales nivel, consigna, error y señal de control. El aspecto de esta ventana se muestra en la siguiente imagen:

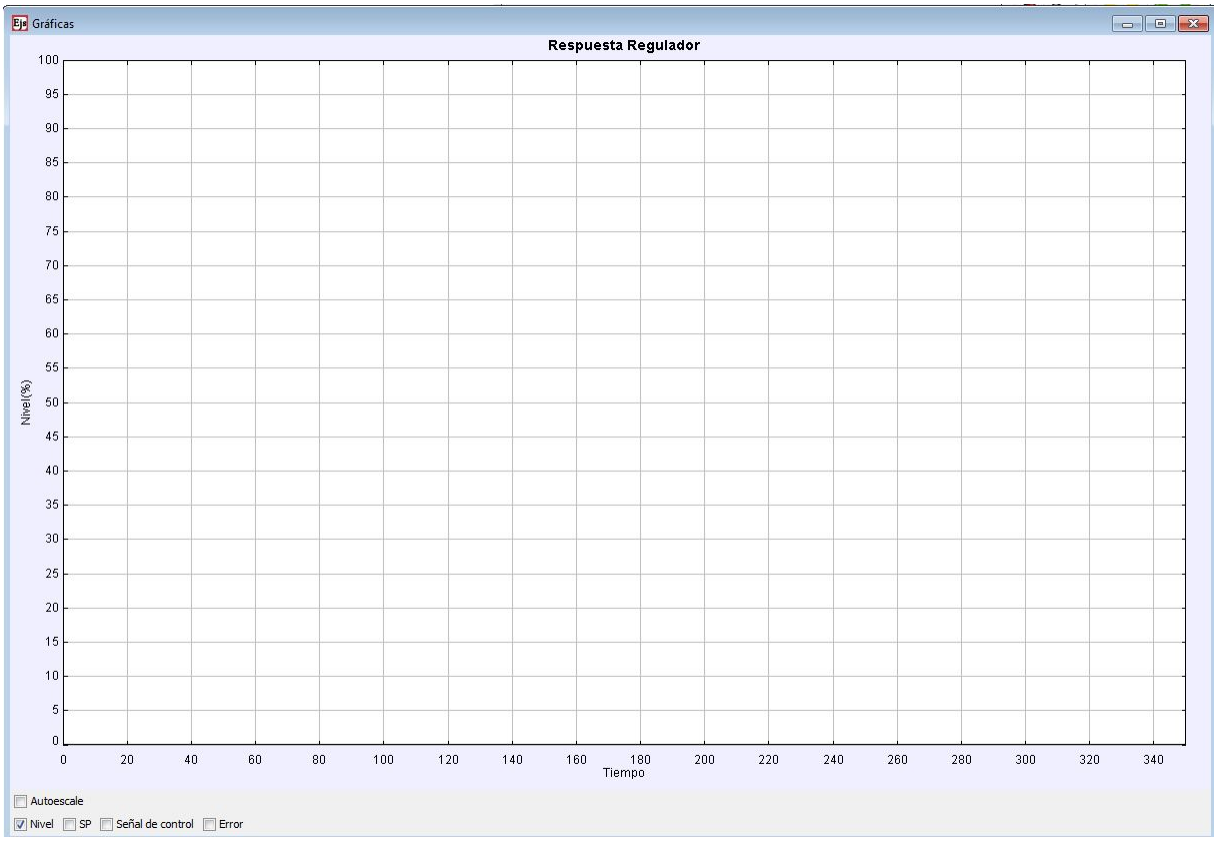



**Figura 8.4.4.1 –** Ventana de Gráficos.

Esta ventana está compuesta por un panel con ejes, de 4 trazas (una por señal) y un menú situado en la parte inferior que permite seleccionar las variables que se deben de mostrar y permite accionar la propiedad Autoescale. La variable que rige la apertura de esta ventana es *graficos_visible=true*.

#### 8.4.4.1. Panel con ejes

El panel con ejes permite la configuración de las siguientes propiedades:

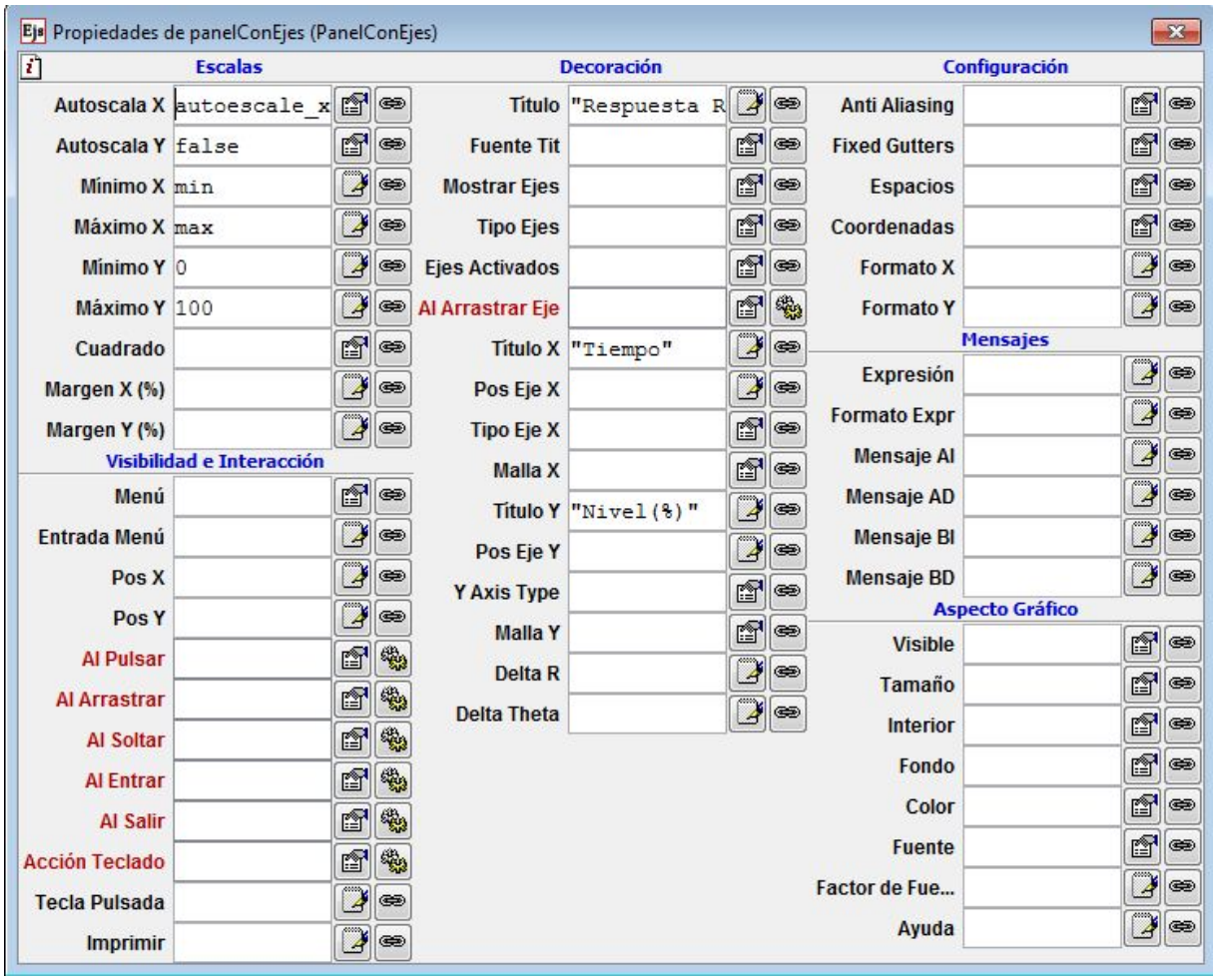



**Figura 8.4.4.2 –** Ventana de propiedades de panel con ejes.

En este caso se configura:

- **Autoscala X:** autoescale_x
- **Autoscala Y:** false
- **Mínimo X:** min
- **Máximo X:** max
- **Mínimo Y:** 0
- **Máximo Y:** 100
- **Título:** Respuesta Regulador
- **Título X:** Tiempo
- **Título Y:** Nivel( %)

#### 8.4.4.2. Trazas

Se introducen cuatro trazas en el elemento panel con ejes para representar la respuesta de las variables nivel, error, set point y señal de control.

Las propiedades que se pueden configurar en el elemento traza se muestran en la imagen siguiente:

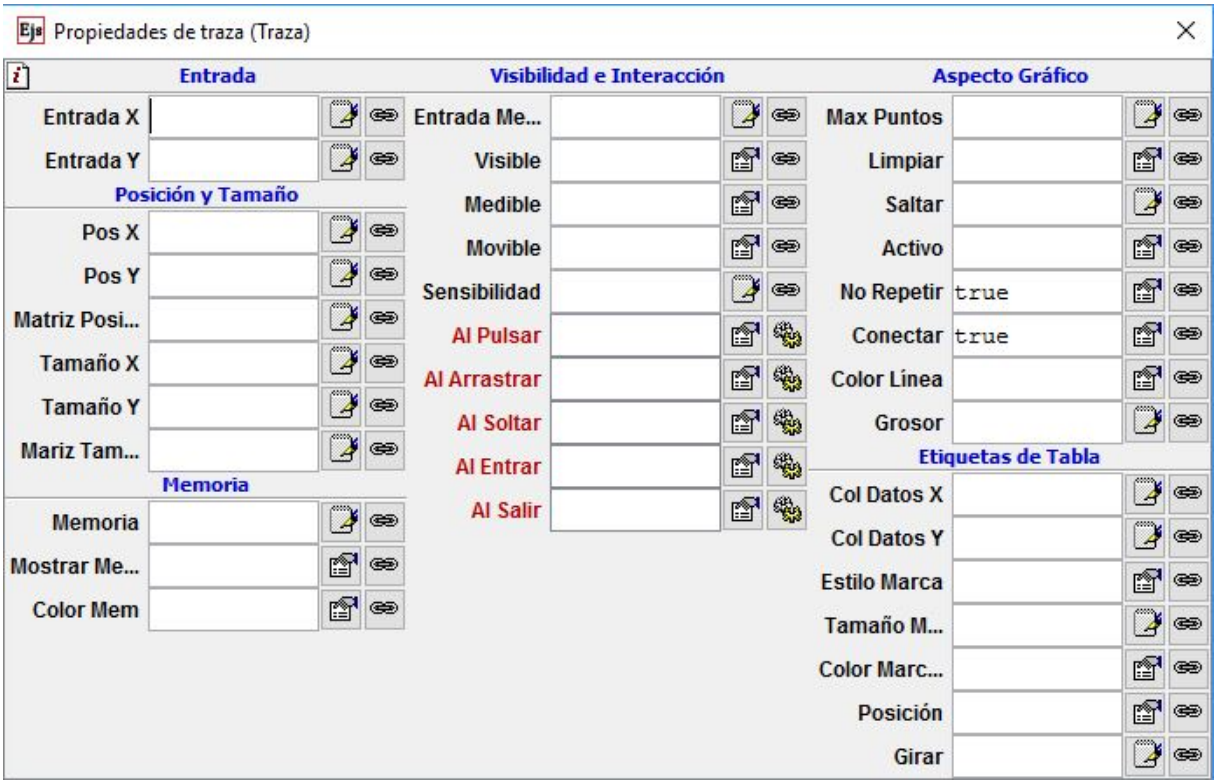



**Figura 8.4.4.3 –** Ventana de propiedades del elemento traza.

La configuración para la traza de la variable nivel es la siguiente:

- **Entrada X:** cont
- **Entrada Y:** nivel_ejs
- **Visible:** nivel_visible
- **Limpiar:** limpiar_gráfica
- **Color Línea:** Red

La traza de la variable set point es la siguiente:

- **Entrada X:** cont
- **Entrada Y:** sp
- **Visible:** SP_visible
- **Limpiar:** limpiar_gráfica

- **Color Línea:** Blue

Los parámetros configurados en la traza de la variable error es la siguiente:

- **Entrada X:** cont
- **Entrada Y:** error_ejs
- **Visible:** error_visible
- **Limpiar:** limpiar_gráfica
- **Color Línea:** Magenta

La traza de la variable señal de control muestra la siguiente configuración:

- **Entrada X:** cont
- **Entrada Y:** senal_control_ejs
- **Visible:** senal_control_visible
- **Limpiar:** limpiar_gráfica
- **Color Línea:** Green

#### 8.4.4.3. Casillas de Selección de trazas

En la parte inferior de la ventana de gráficos se encuentra cuatro casillas de verificación que rigen la visibilidad de las trazas.

La pantalla de configuración de las casillas de verificación se muestra a continuación:


**Figura 8.4.4.4 –** Ventana de propiedades de la casilla de verificación.

La configuración que se realiza para la casilla de verificación de la variable nivel es la siguiente:

- **Variable:** nivel_visible
- **Seleccion:** true

- **Texto:** Nivel
- **Acción Si:** nivel_visible=true
- **Acción No:** nivel_visible=false

La casilla de verificación de la variable set point presenta la siguiente configuración:

- **Variable:** sp_visible
- **Seleccion:** false
- **Texto:** SP
- **Acción Si:** sp_visible=true
- **Acción No:** sp_visible=false

La configuración de la casilla de verificación de la variable señal de control es la siguiente:

- **Variable:** senal_control_visible
- **Seleccion:** false
- **Texto:** Señal de Control
- **Acción Si:** senal_control_visible=true
- **Acción No:** senal_control_visible=false

La casilla de verificación de la variable error se muestra a continuación:

- **Variable:** error_visible
- **Seleccion:** false
- **Texto:** Error
- **Acción Si:** error_visible=true
- **Acción No:** error_visible=false

#### 8.4.4.4. Casilla de Selección de Autoscale

En la parte inferior se encuentra una casilla de selección de Autoscale. Con la opción Autoscale seleccionada, el Eje X del gráfico será variable, mientras que si no se selecciona este eje será fijo.

La pantalla de configuración de la casilla de verificación se muestra en la imagen 8.4.4.4.

La configuración de la casilla de selección Autoscale es la siguiente:

- **Variable:** autoescale_x
- **Selección:** false

- **Texto:** Autoscale
- **Acción Si:** autoescale_x=true
- **Acción No:** autoescale_x=false

La gestión de esta función se realiza en el apartado de relaciones fijas. Esta gestión consiste en ubicar los límites del eje X cuando se de-selecciona la opción Autoscale.

El código de esta función es el siguiente:

```
//Ajuste del autoscale
if(cont==max && autoescale_x==false){,caption=Autoescale
min=min+350;
max=max+350;
}
```

## 8.5. Funcionamiento de Remote EUP

Al abrir Remote EUP se muestra la pantalla principal mostrada en la figura 8.4.1.1. En primer lugar se inicia una sesión de Matlab, para ello accedemos a Conexión – Conectar.

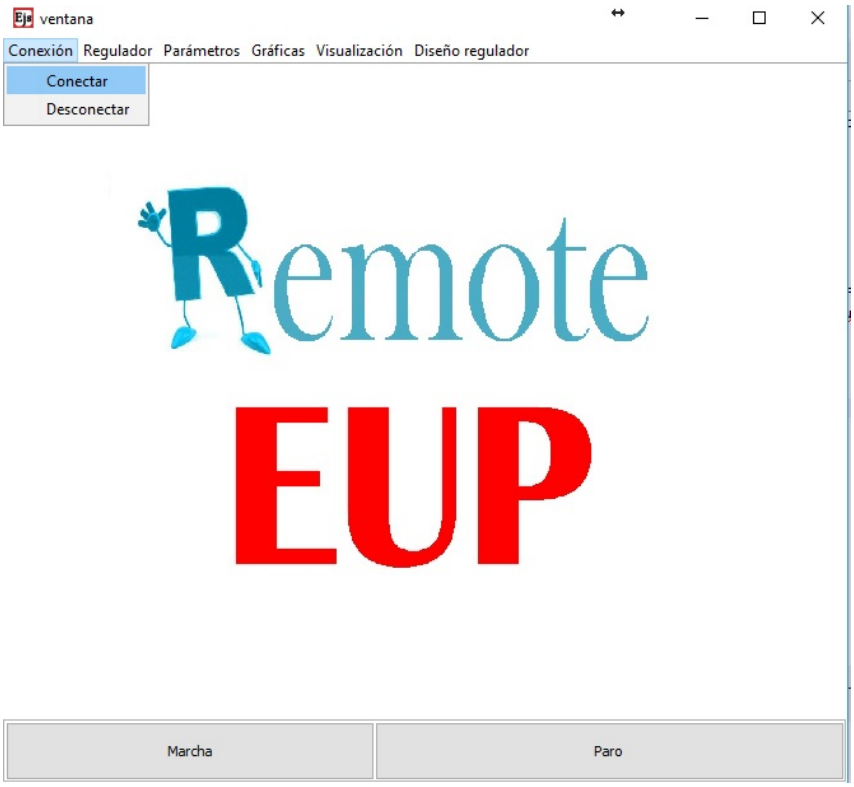



**Figura 8.5.0.1 –** Inicio de sesión de Matlab.

Tras el inicio de sesión en Matlab, se obtienen los parámetros del regulador por el método de Relay Feedback. Se accede a Diseño Regulador – Cadena Cerrada – Relay - Feedback.

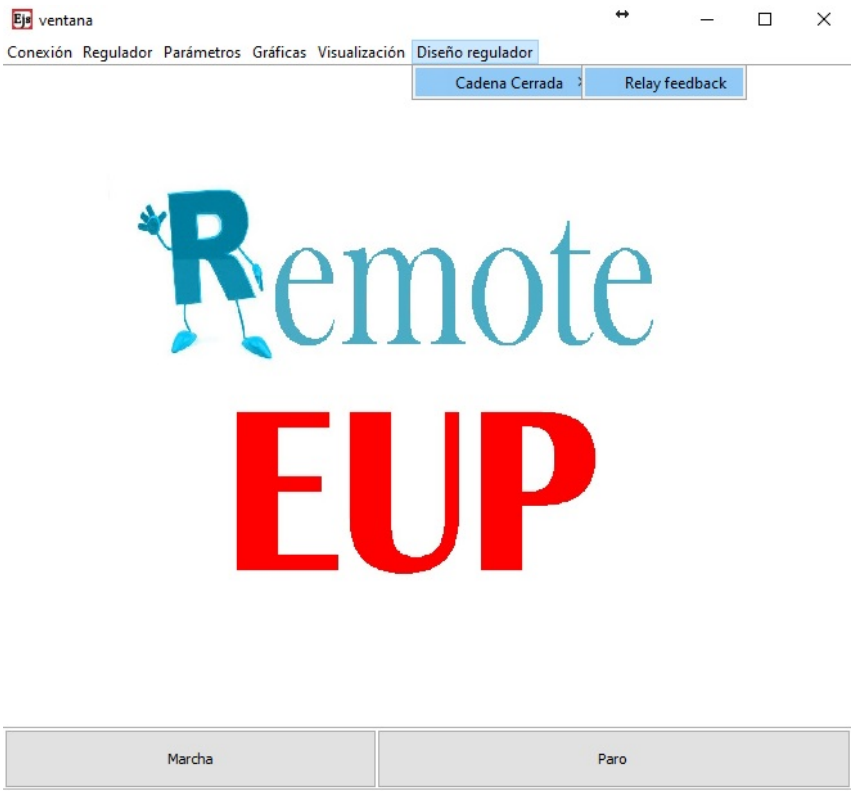



**Figura 8.5.0.2 –** Selección del método de Relay - Feedback.

A continuación se insertan los parámetros para la realización de este método. Para ello se accede a Parámetros – Parámetros Relay - Feedback.

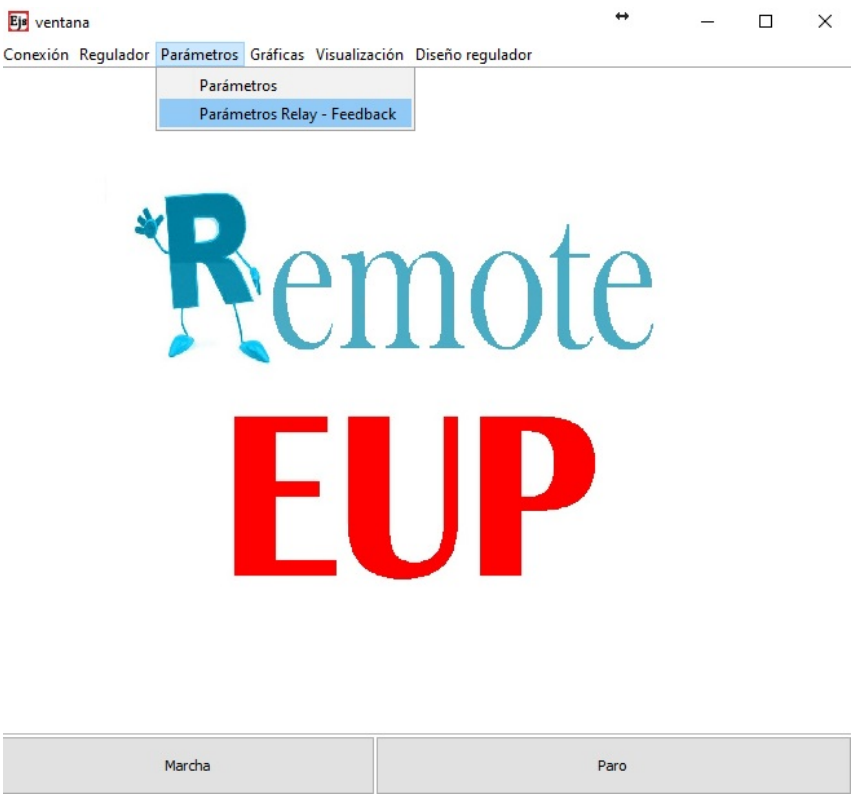



**Figura 8.5.0.3 –** Selección de los parámetros del Método Relay - Feedback.

Se muestra la ventana de inserción de los parámetros de Relay - Feedback y se introducen unos valores de 50 para punto de trabajo y de 10 para el ancho de ventana.

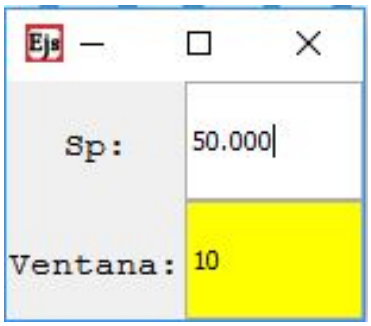



**Figura 8.5.0.4 –** Parámetros del método de Relay - Feedback.

Una vez configurado el método de Relay - Feedback se pone en funcionamiento el sistema pulsando el botón de MARCHA en la ventana principal. En este momento comienza la ejecución del método de Relay - Feedback. Para parar la simulación se debe de pulsar el botón de PARO en la ventana principal.

La respuesta que se obtiene en el ordenador local es la siguiente:

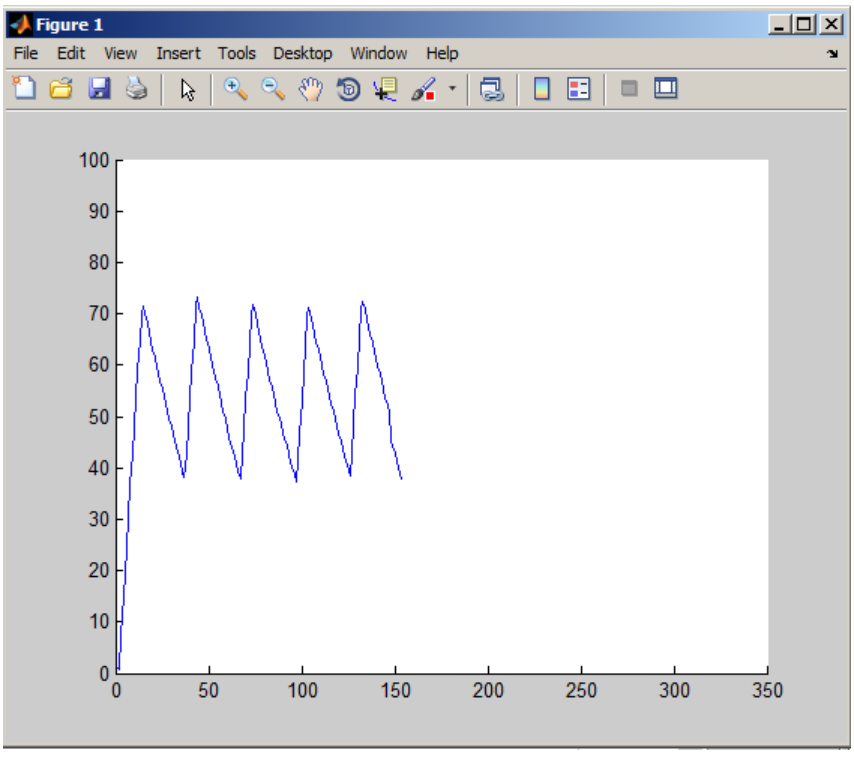



**Figura 8.5.0.5 –** Respuesta del método de Relay - Feedback en Matlab.

La respuesta que se obtiene en el ordenador remoto se muestra en la siguiente imagen. Remote EUP permite obtener las coordenadas de un punto de la gráfica situando el cursor encima y realizando click. Se acotan los puntos de interés para la obtención de los parámetros $T_c$ y $a$.

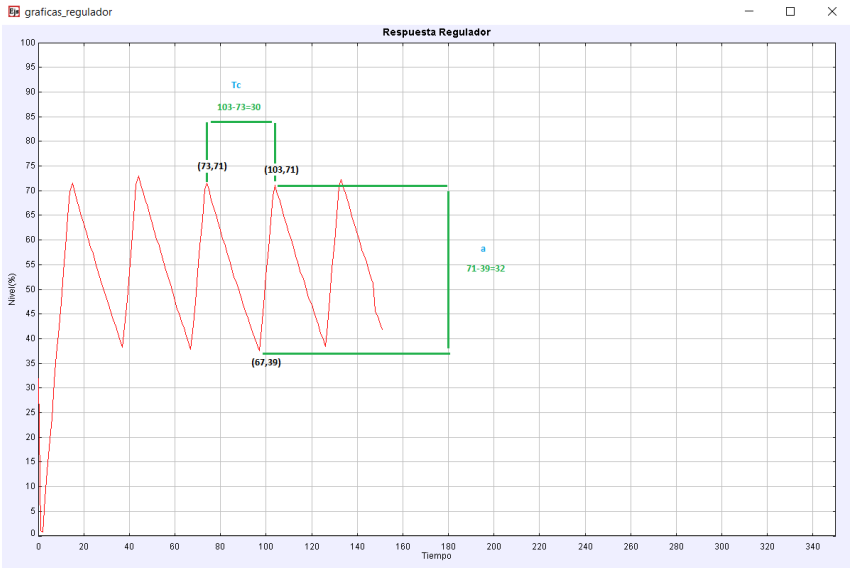



**Figura 8.5.0.6 –** Respuesta del método de Relay - Feedback en Remote EUP.

Tras estudiar la respuesta del método de Relay - Feedback se obtienen los siguientes parámetros:

| Parámetro | Valor |
|---|---|
| $T_c$ | 30 |
| $a$ | 32 |

**Tabla 8.5.0.1 –** Parámetros obtenidos con el método de Relay - Feedback

Haciendo uso de la ecuación 7.2.3.1, se halla el parámetro $K_c$.

$$K_c = \frac{4 * 50}{\pi * \sqrt{32^2 - 10^2}} = 2{,}094$$

Mediante el uso de la ecuación 7.2.4.1, se obtienen los parámetros del regulador correspondientes a la aproximación de Ziegler - Nichols.

| Parámetro | Valor |
|---|---|
| $k$ | 1,25 |
| $Ti$ | 15 |
| $Td$ | 3,75 |

**Tabla 8.5.0.2 –** Valor de los parámetros del regulador mediante Ziegler - Nichols

Los valores de los parámetros de la aproximación de Ziegler - Nichols No Overshoot se obtienen a partir de la ecuación 7.2.4.3.

| Parámetro | Valor |
|---|---|
| $k$ | $0{,}42$ |
| $Ti$ | $30$ |
| $Td$ | $10$ |

**Tabla 8.5.0.3 –** Valor de los parámetros del regulador mediante Ziegler - Nichols No Overshoot

Haciendo uso de la ecuación 7.2.4.2 se obtienen los parámetros mediante Ziegler - Nichols Some Overshoot.

| Parámetro | Valor |
|---|---|
| $k$ | $0{,}69$ |
| $Ti$ | $15$ |
| $Td$ | $10$ |

**Tabla 8.5.0.4 –** Valor de los parámetros del regulador mediante Ziegler - Nichols Some Overshoot

Los parámetros correspondientes a la aproximación de Tyreus - Luyben se calculan a partir de la ecuación 7.2.4.4.

| Parámetro | Valor |
|---|---|
| $k$ | $0{,}94$ |
| $Ti$ | $66$ |
| $Td$ | $4{,}76$ |

**Tabla 8.5.0.5 –** Valor de los parámetros del regulador mediante Tyreus - Luyben

Una vez obtenidos los parámetros propios del regulador, se procede a su implantación. Para ello se debe de acceder a Regulador y elegir el regulador a implementar.

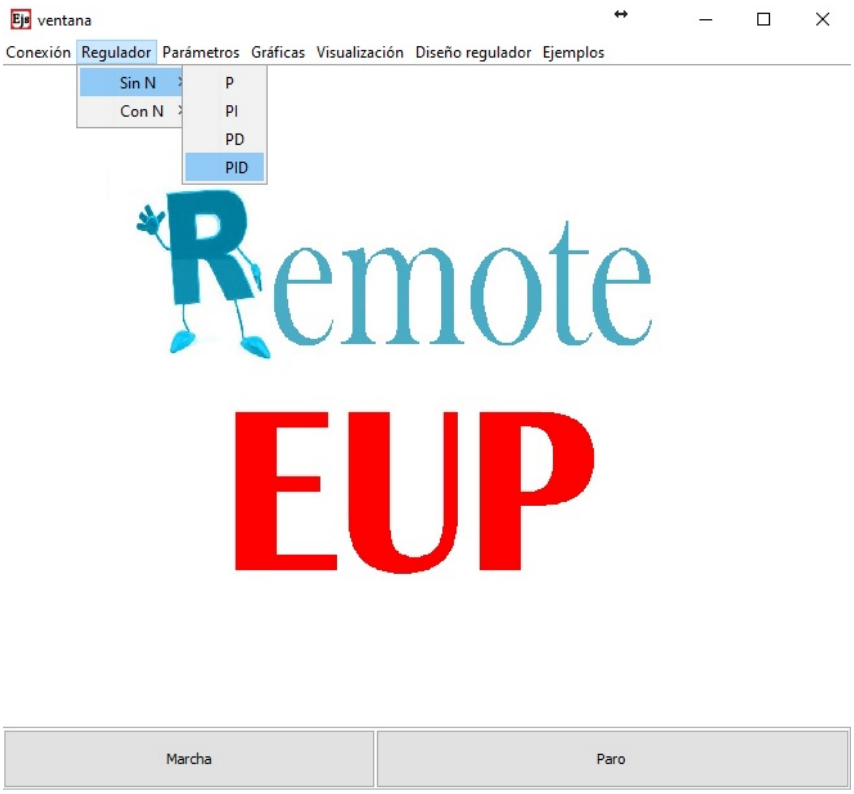



**Figura 8.5.0.7 –** Selección de regulador a implementar.

A continuación se accede a Parámetros – Parámetros y se introducen los valores de los parámetros del regulador. En este caso se introducen los parámetros correspondiente a Ziegler - Nichols.

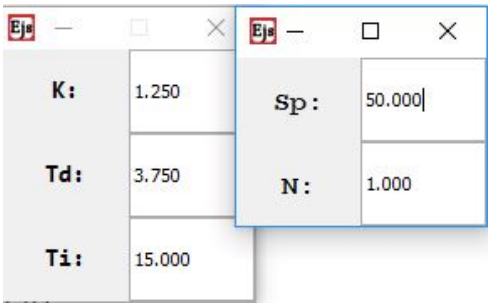



**Figura 8.5.0.8 –** Introducción de parámetros del regulador.

Tras introducir los parámetros, el regulador está listo para su puesta en funcionamiento. Se pulsa el botón de MARCHA ubicado en la pantalla de inicio y se implementa el regulador.

Durante el funcionamiento del regulador se puede ver el estado físico de la planta en tiempo real accediendo a Visualización – Visualización. Se abre una página en el navegador de internet predeterminado del usuario y se muestra la imagen de la web cam.

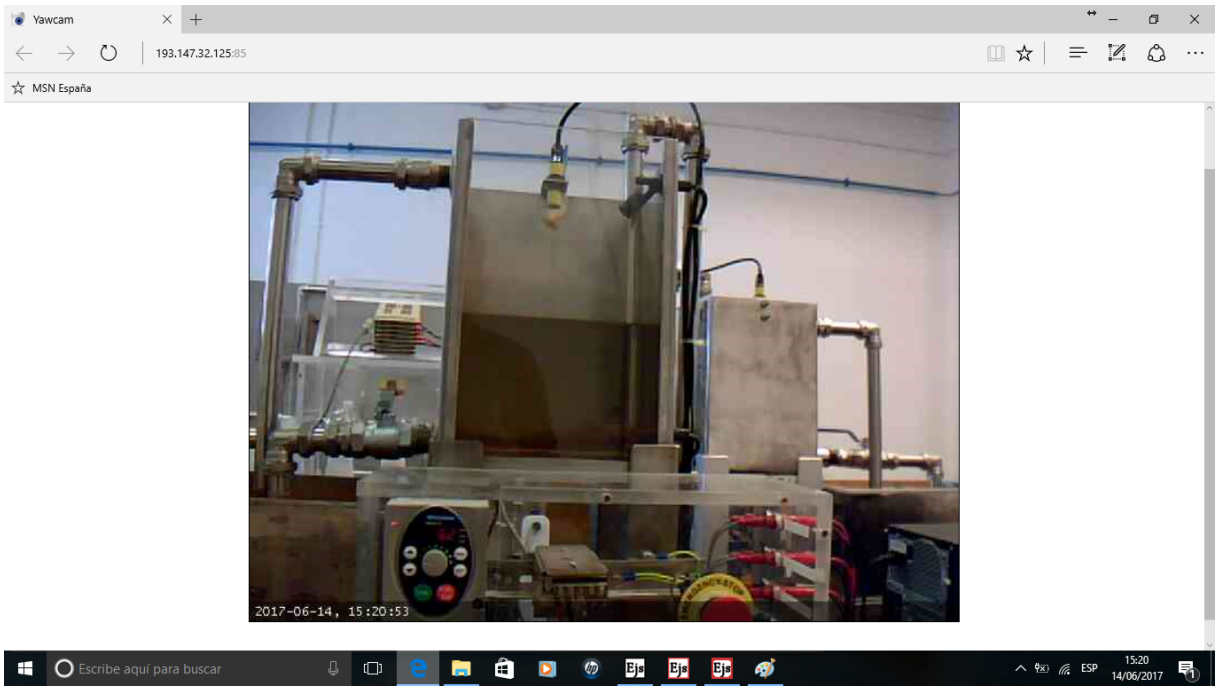



**Figura 8.5.0.9 –** Imagen de la planta en tiempo real.

Para ver gráficamente la respuesta del regulador se debe de acceder a Gráficas – Mostrar Gráfica.

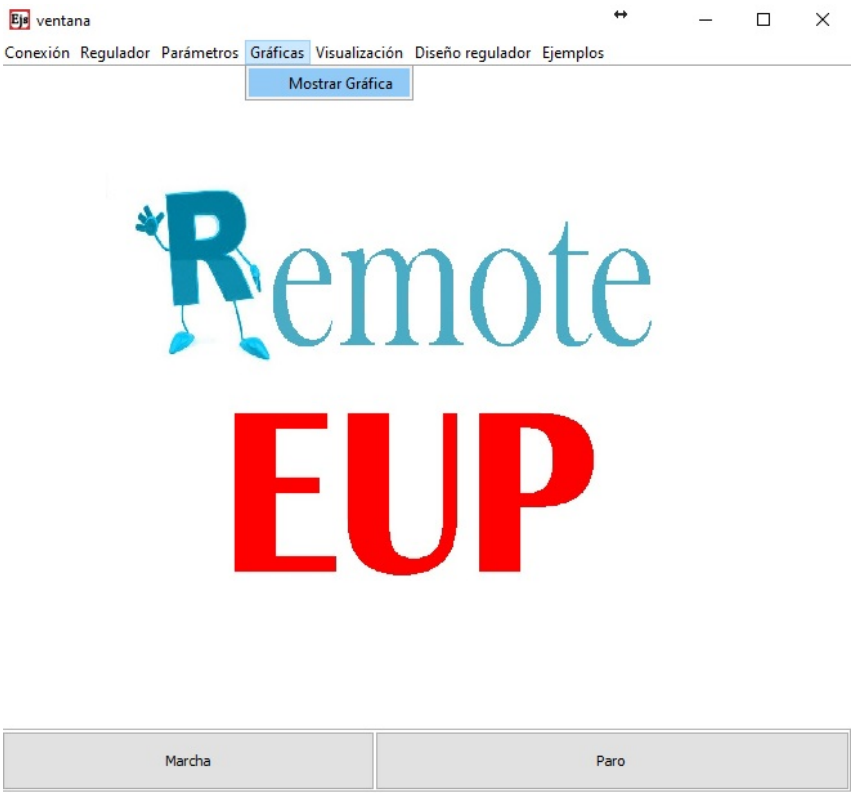



**Figura 8.5.0.10 –** Abrir gráfica.

Si se desea parar el regulador implementado se debe de pulsar sobre el botón de PARO que está ubicado en la ventana principal. Cuando se para Remote EUP, se inicializan las variables y está preparado para la implementación de otro regulador. La gráfica del regulador se mantiene hasta que se pulse MARCHA para implantar otro nuevo controlador.

### 8.5.1. Respuesta de los Reguladores sin corrección

La respuesta que se ha obtenido haciendo uso de la **aproximación de Ziegler - Nichols** es la siguiente:

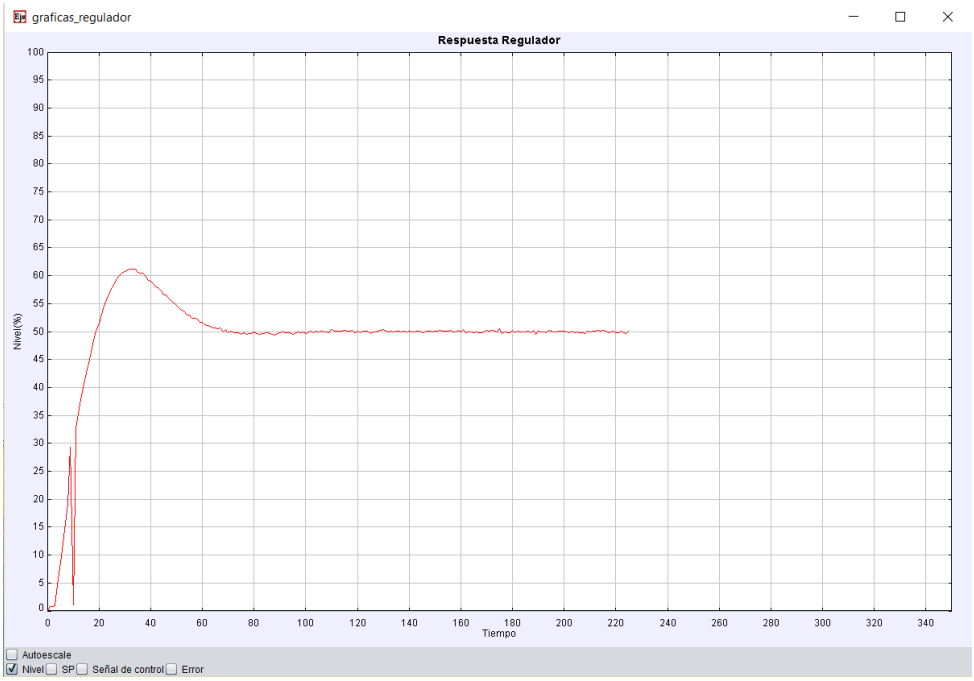



**Figura 8.5.1.1 –** Respuesta regulador Ziegler - Nichols en Remote EUP.

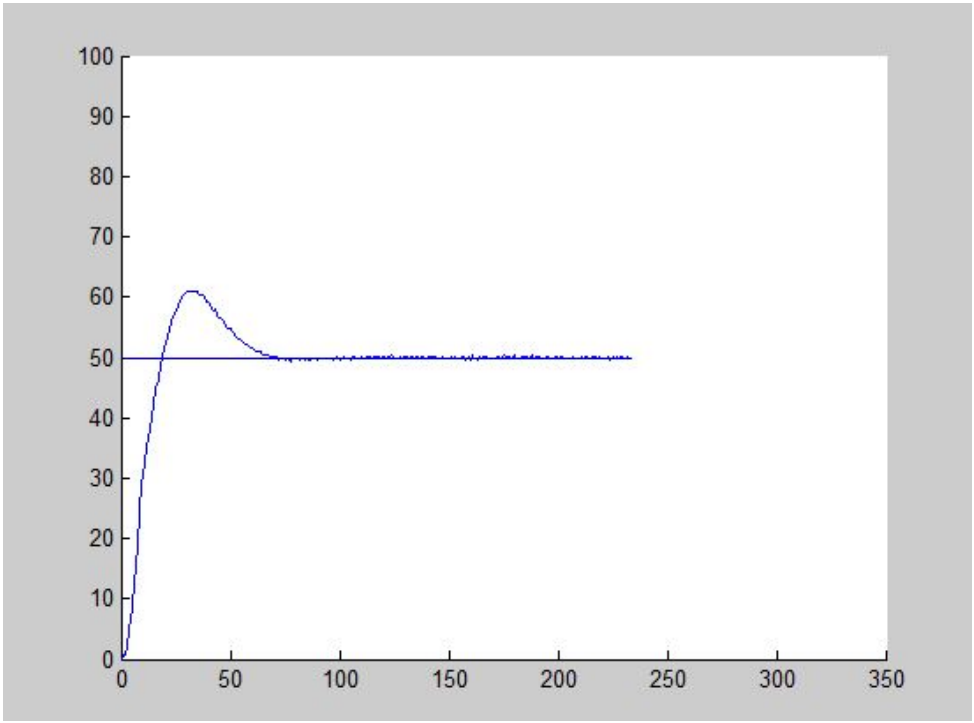


**Figura 8.5.1.2 –** Respuesta regulador Ziegler - Nichols en Matlab.

La respuesta que se ha obtenido haciendo uso de la **aproximación de Ziegler - Nichols No Overshoot** es la siguiente:

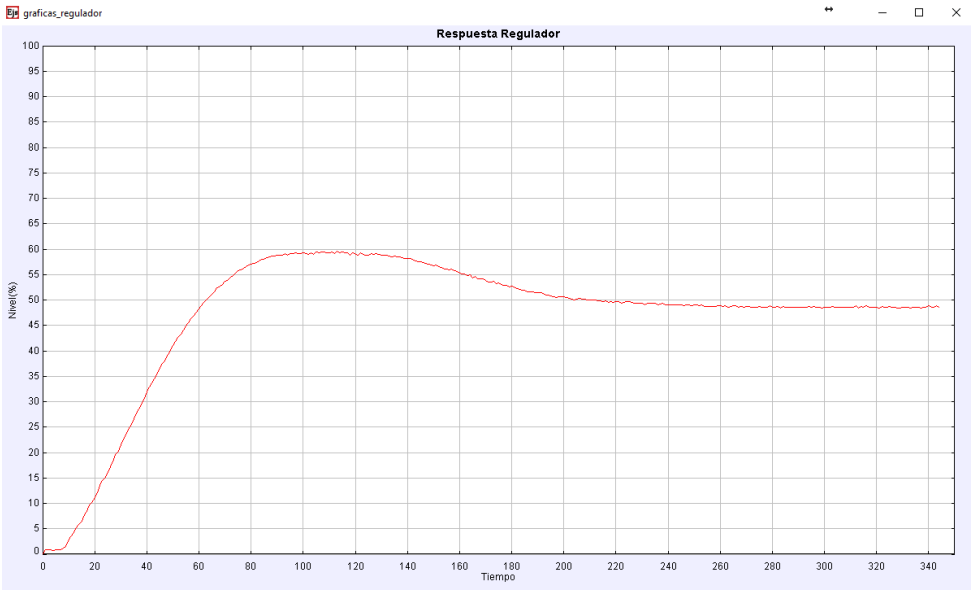



**Figura 8.5.1.3 –** Respuesta regulador Ziegler - Nichols No Overshoot en Remote EUP.

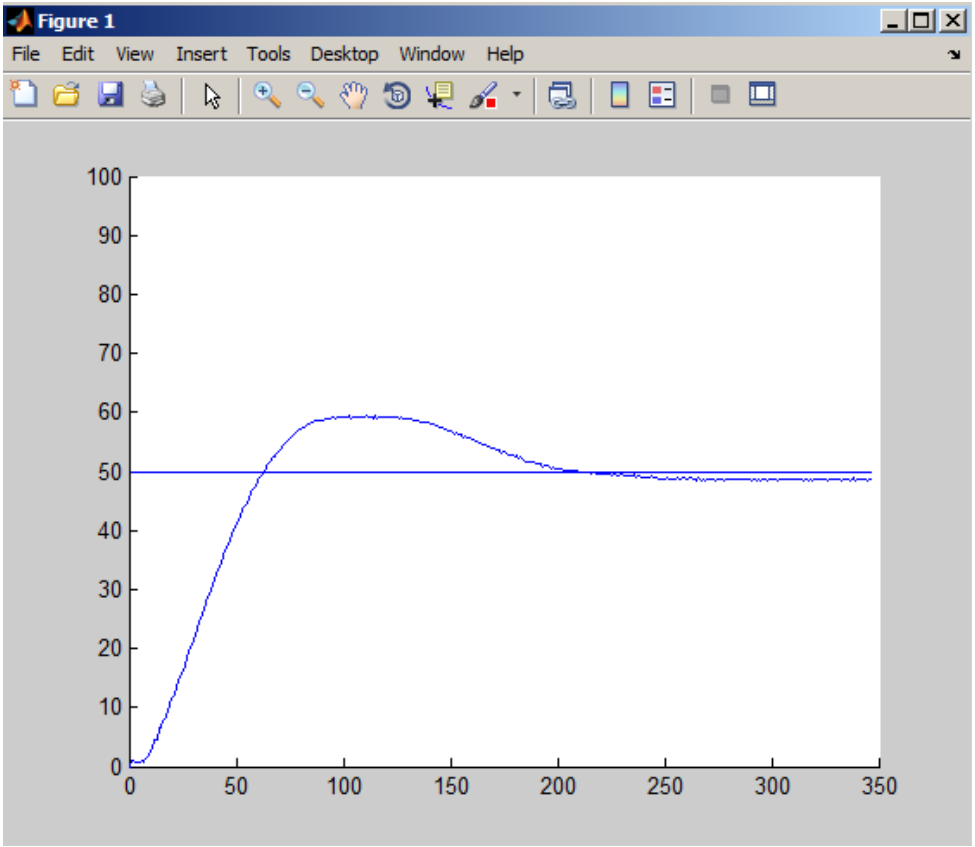



**Figura 8.5.1.4 –** Respuesta regulador Ziegler - Nichols No Overshoot en Matlab.

La respuesta que se ha obtenido haciendo uso de la **aproximación de Ziegler - Nichols Some Overshoot** es la siguiente:

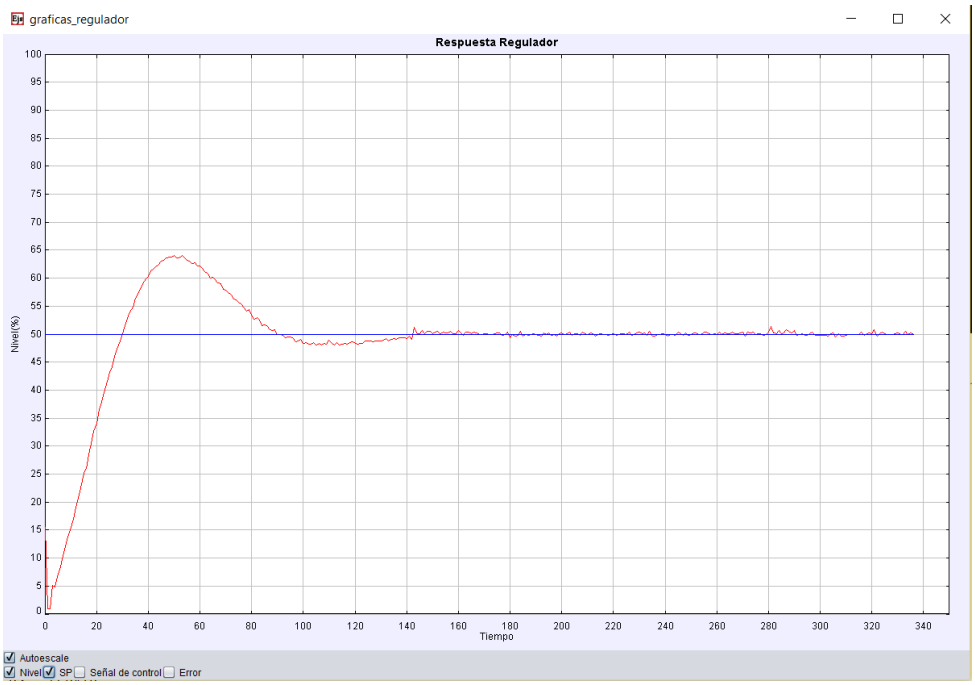



**Figura 8.5.1.5 –** Respuesta regulador Ziegler - Nichols Some Overshoot en Remote EUP.

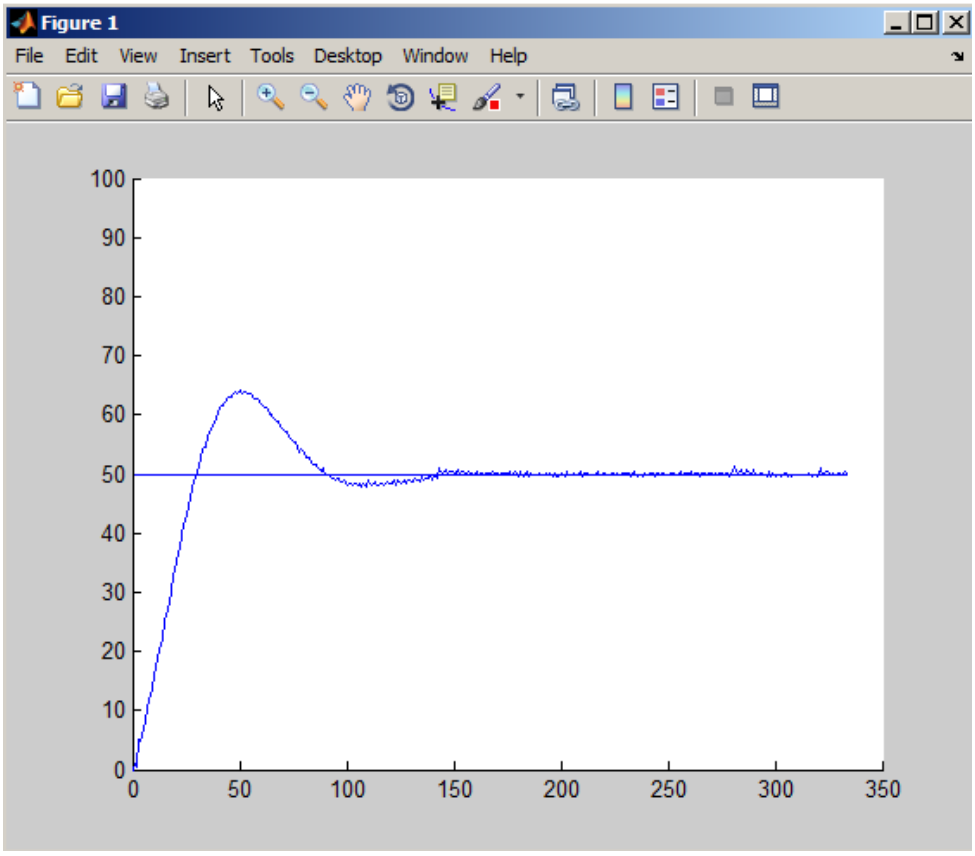



**Figura 8.5.1.6 –** Respuesta regulador Ziegler - Nichols Some Overshoot en Matlab.

La respuesta que se ha obtenido haciendo uso de la **aproximación de Tydeus - Luyben** es la siguiente:

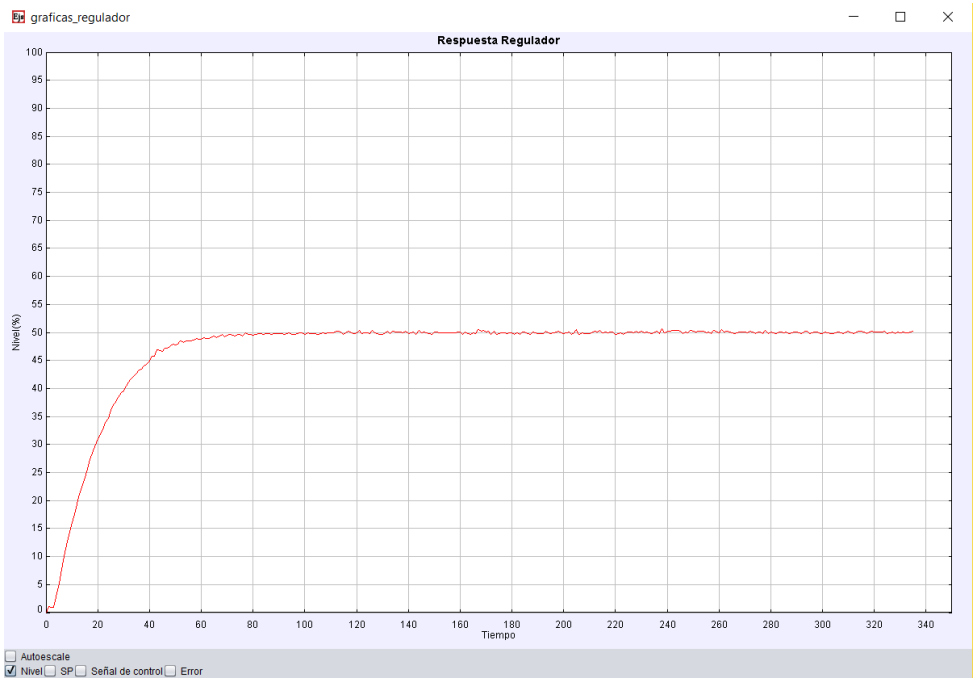



**Figura 8.5.1.7 –** Respuesta regulador Tydeus - Luyben en Remote EUP.

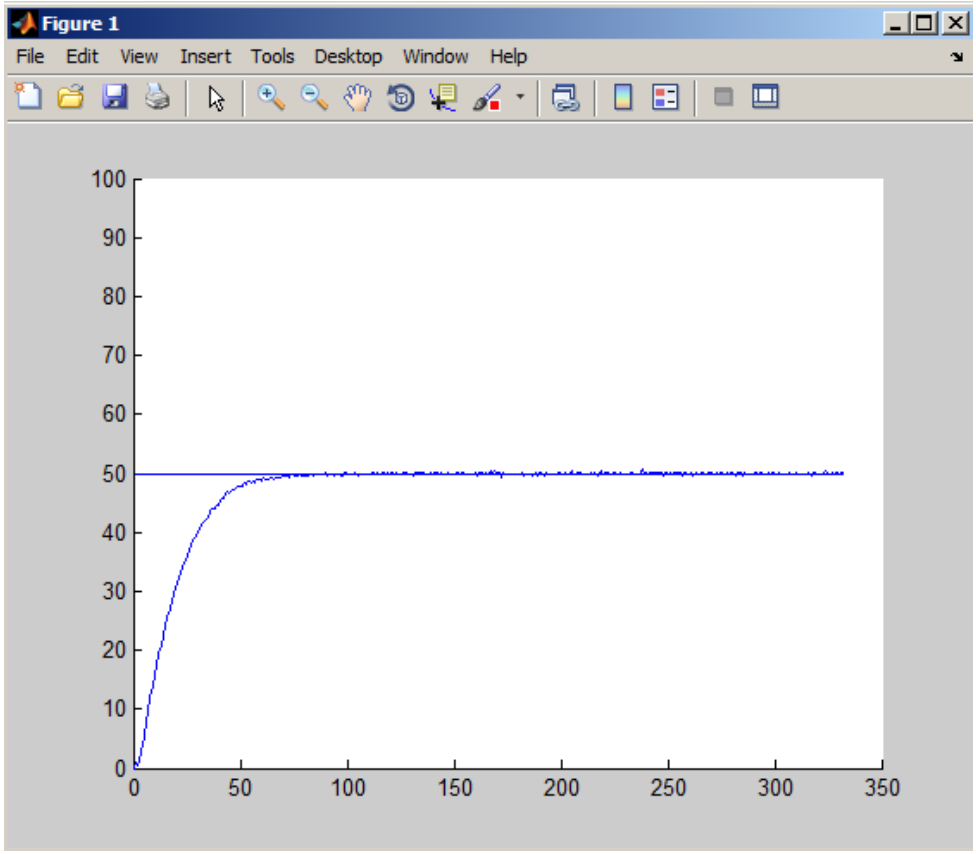



**Figura 8.5.1.8 –** Respuesta regulador Tydeus - Luyben en Matlab.

### 8.5.2. Respuesta de los reguladores con corrección

Para los reguladores con corrección se ha utilizado una N de 0.01, con un valor tan bajo reducimos la acción derivativa y se consigue una respuesta buena sobre el sistema.

La respuesta que se ha obtenido haciendo uso de la **aproximación de Ziegler - Nichols** es la siguiente:

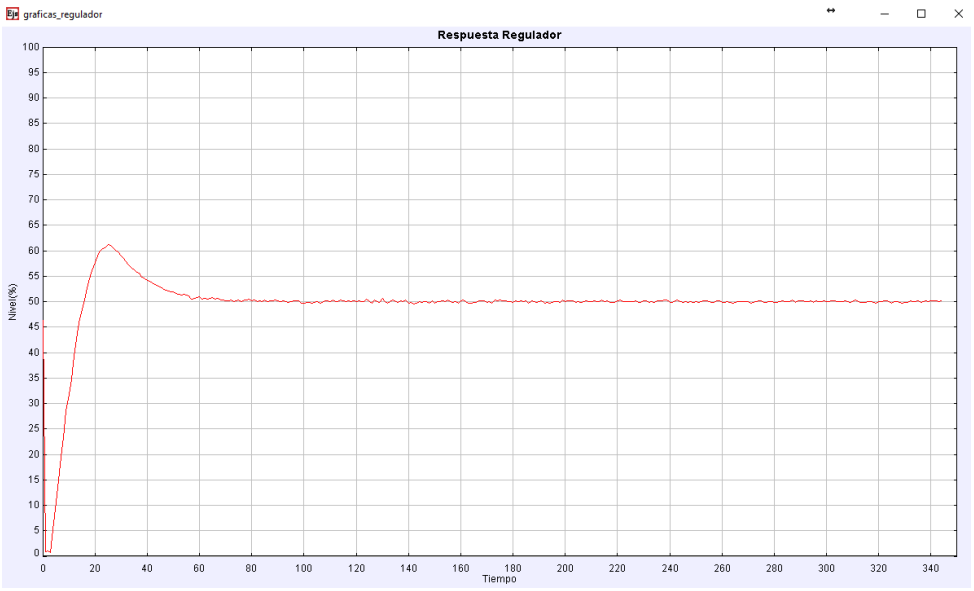



**Figura 8.5.2.1 –** Respuesta regulador Ziegler - Nichols con corrección en Remote EUP.

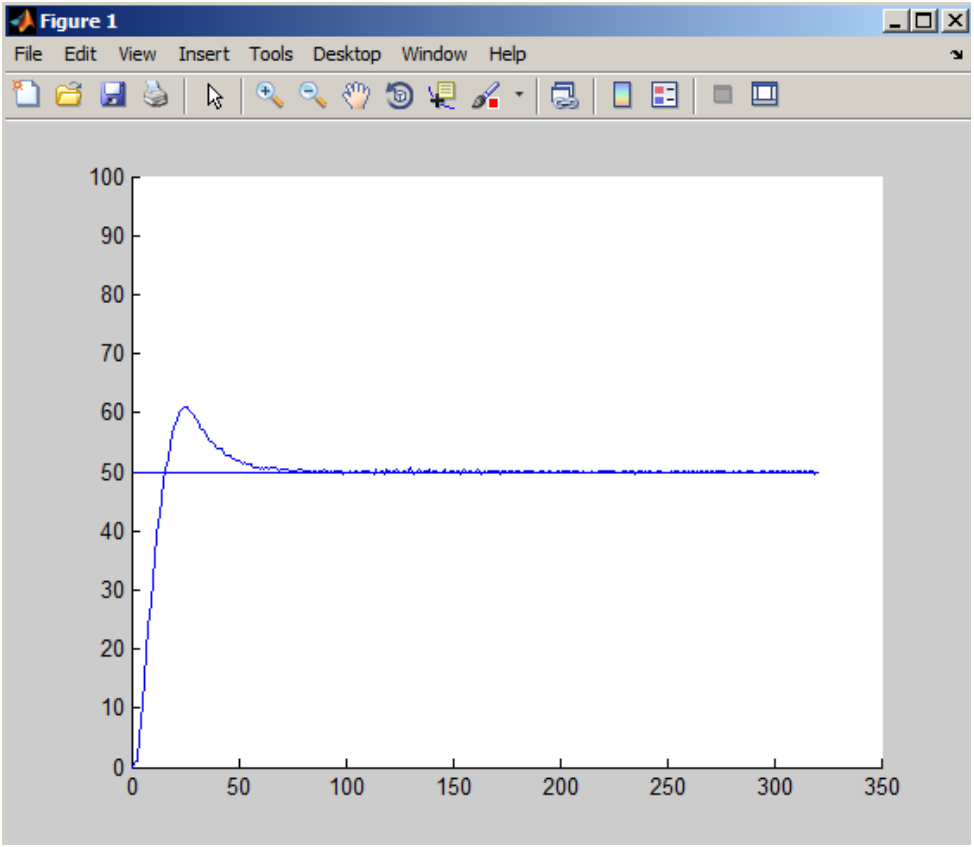



**Figura 8.5.2.2 –** Respuesta regulador Ziegler - Nichols con corrección en Matlab.

La respuesta que se ha obtenido haciendo uso de la **aproximación de Ziegler - Nichols No Overshoot** es la siguiente:

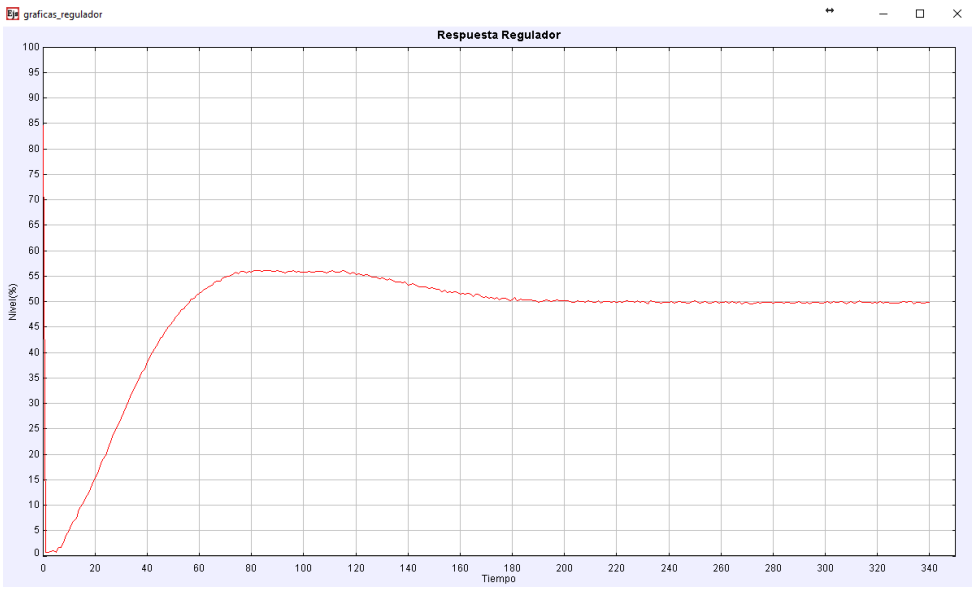



**Figura 8.5.2.3 –** Respuesta regulador Ziegler - Nichols No Overshoot con corrección en Remote EUP.

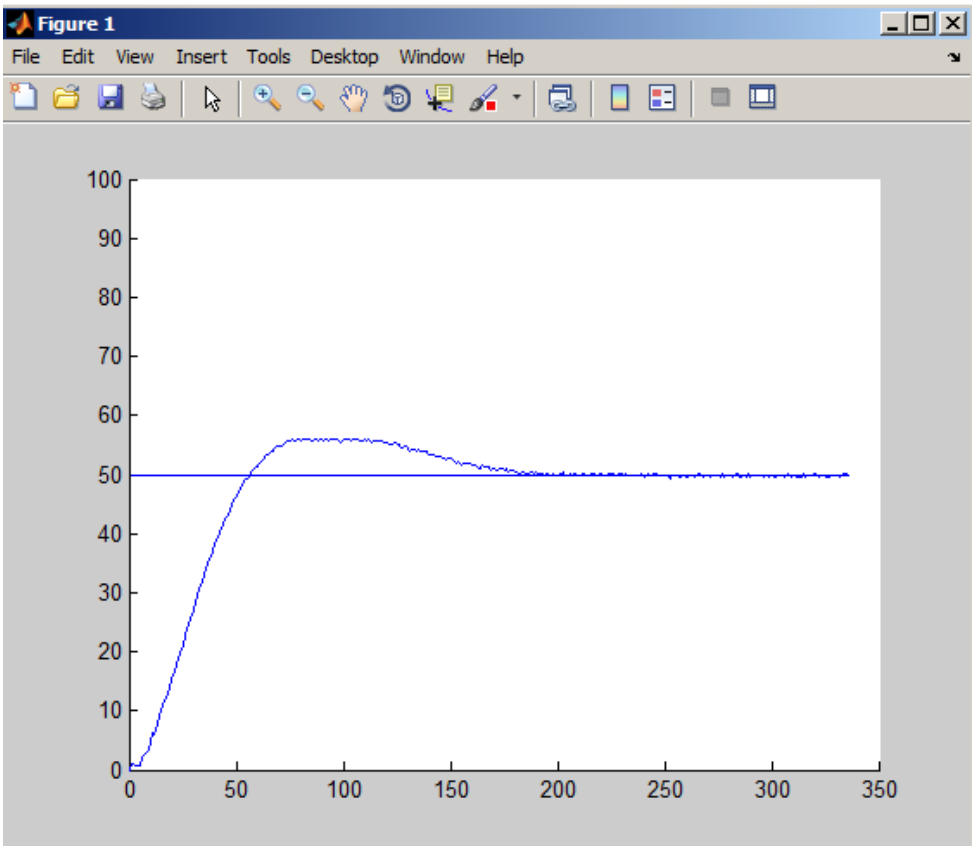



**Figura 8.5.2.4 –** Respuesta regulador Ziegler - Nichols No Overshoot con corrección en Matlab.

La respuesta que se ha obtenido haciendo uso de la **aproximación de Ziegler - Nichols Some Overshoot** es la siguiente:

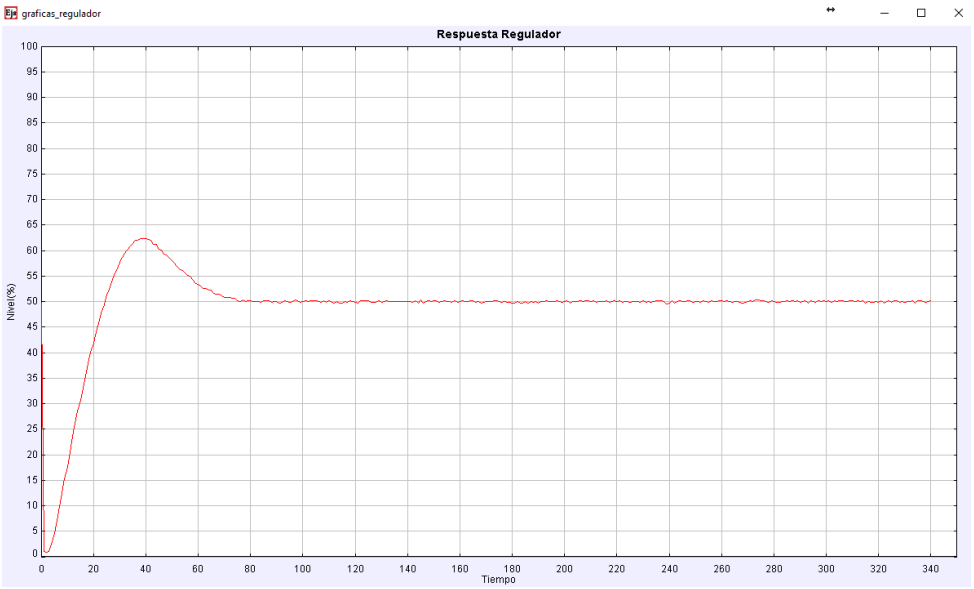



**Figura 8.5.2.5 –** Respuesta regulador Ziegler - Nichols Some Overshoot con corrección en Remote EUP.

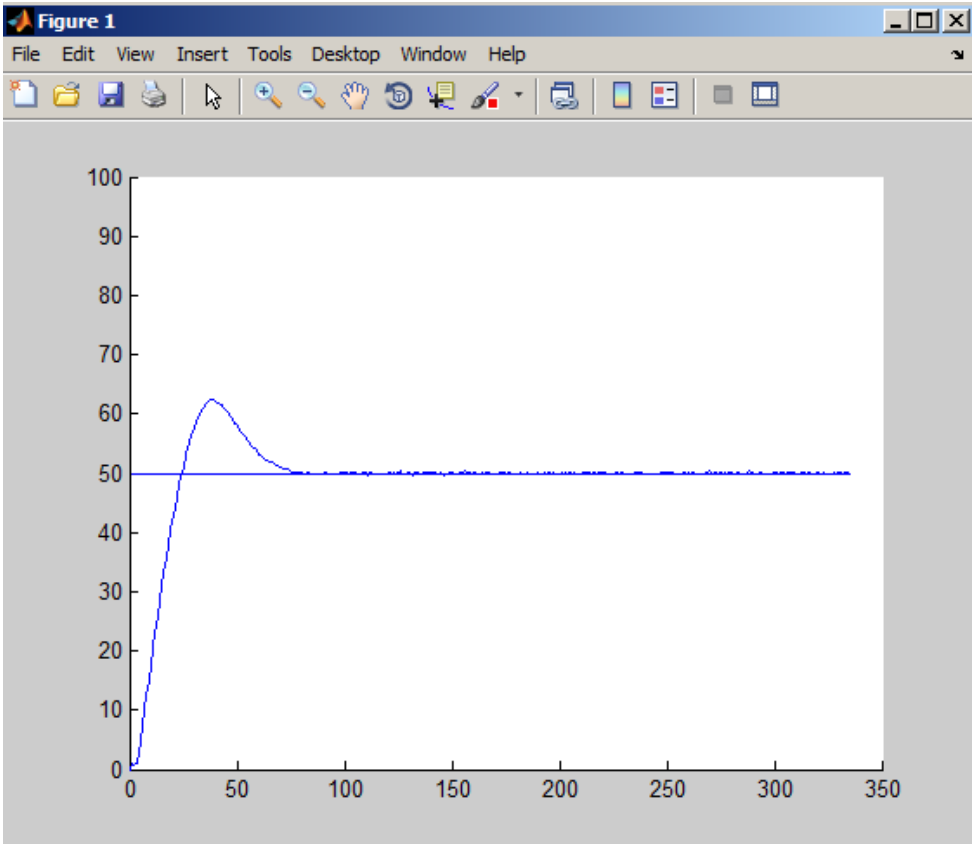



**Figura 8.5.2.6 –** Respuesta regulador Ziegler - Nichols Some Overshoot con corrección en Matlab.

La respuesta que se ha obtenido haciendo uso de la **aproximación de Tydeus - Luyben** es la siguiente:

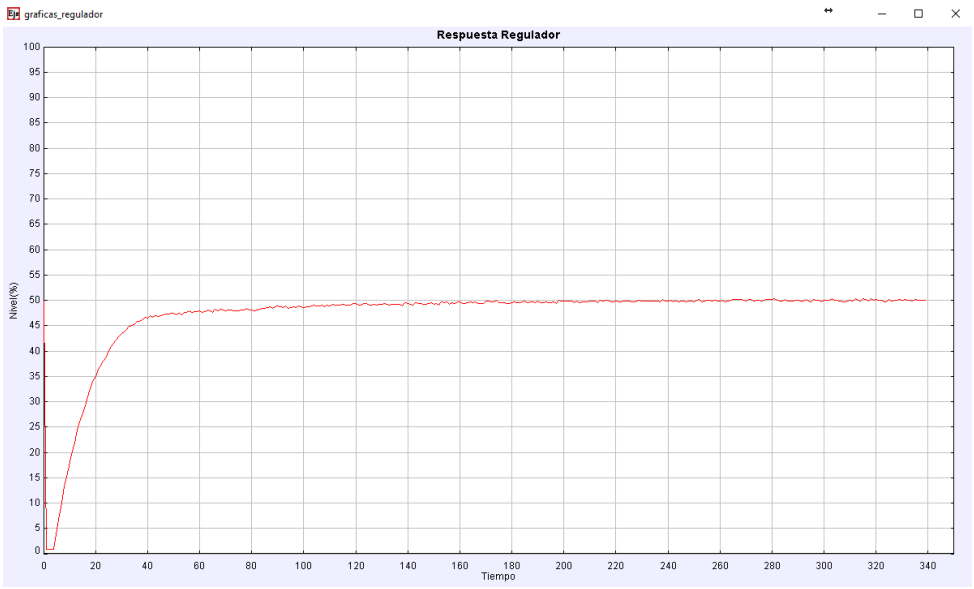



**Figura 8.5.2.7 –** Respuesta regulador Tydeus - Luyben con corrección en Remote EUP.

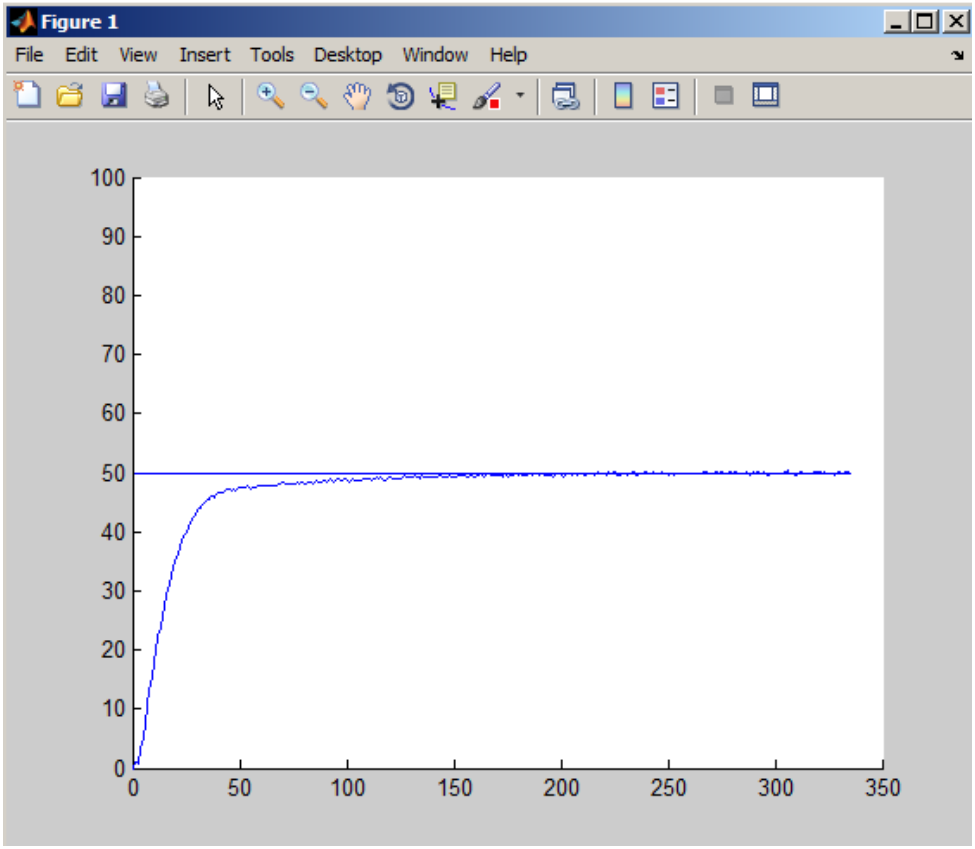



**Figura 8.5.2.8 –** Respuesta regulador Tydeus - Luyben con corrección en Matlab.

## 8.6. Base de datos

En el desarrollo de Remote EUP se ha decidido hacer uso de matrices fijas en Matlab por motivos de eficiencia, por lo que solo se guardan los últimos 350 puntos de la simulación. Para no perder todos los datos del proceso, se almacena una base de datos en el ordenador local y una base de datos en el ordenador remoto. Esta base de datos presenta un formato *.csv*, por lo que se puede abrir con cualquier editor de texto. Una vez finalizada una simulación se puede acceder a esta base de datos y manejar los datos de la simulación. A continuación se muestra como ejemplo la representación en el programa Microsoft Excel de la respuesta del regulador obtenido mediante la aproximación de Ziegler - Nichols.

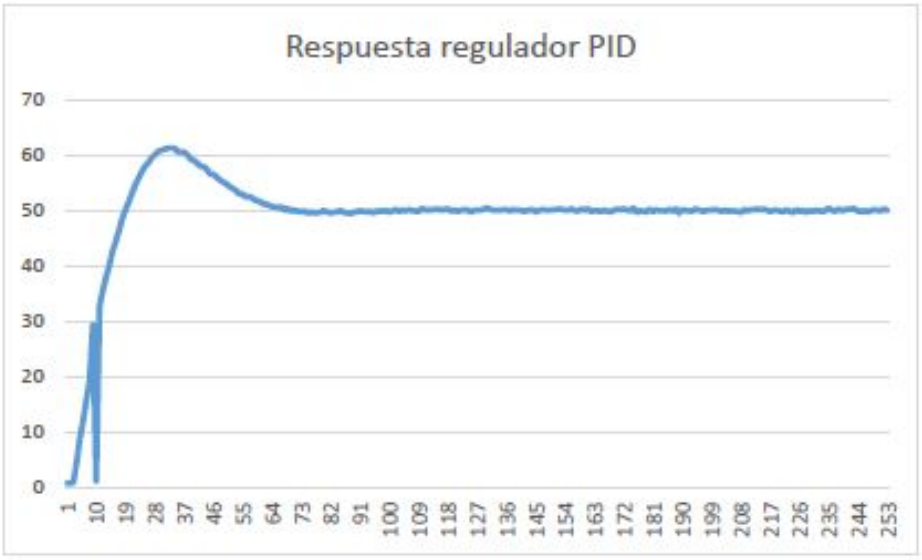



**Figura 8.6.0.1 –** Respuesta regulador Ziegler - Nichols en Excel.

## 8.7. Tiempo de muestreo

El primer principio para dar un regulador como válido es que el tiempo de muestreo para su implantación debe de ser constante. En este caso se ha seleccionado el tiempo de muestreo de un segundo. Este tiempo de muestreo es suficiente para realizar la regulación sobre la planta de nivel y es lo suficientemente grande como para no saturar la red. Se imprime en la pantalla de Matlab el tiempo de muestreo en cada iteración mediante la orden *tic* y *toc*.

```
Elapsed time is 1.013444 seconds.
Elapsed time is 1.003801 seconds.
Elapsed time is 1.002291 seconds.
Elapsed time is 1.003555 seconds.
Elapsed time is 1.006745 seconds.
Elapsed time is 1.008295 seconds.
Elapsed time is 0.995804 seconds.
Elapsed time is 1.006607 seconds.
Elapsed time is 1.003813 seconds.
Elapsed time is 1.000191 seconds.
Elapsed time is 0.994585 seconds.
Elapsed time is 0.997539 seconds.
Elapsed time is 1.005689 seconds.
Elapsed time is 1.003272 seconds.
Elapsed time is 1.012208 seconds.
Elapsed time is 0.996327 seconds.
Elapsed time is 1.000336 seconds.
Elapsed time is 0.998283 seconds.
Elapsed time is 0.997534 seconds.
Elapsed time is 1.005171 seconds.
Elapsed time is 1.007279 seconds.
Elapsed time is 1.009117 seconds.
Elapsed time is 1.014054 seconds.
Elapsed time is 1.004140 seconds.
```

**Figura 8.7.0.1 –** Tiempo de muestreo de 1 segundo.

Como se puede observar, el tiempo de muestreo es constante y de aproximadamente 1 segundo.

## 8.8. Gestión de Usuarios en Remote EUP

El control de la planta de nivel del laboratorio de optimización y control puede ser realizado por un único usuario a través de Remote EUP. Esta aplicación de control permite el inicio de varias sesiones de Matlab a la vez, pero no se puede gestionar una misma planta desde varias sesiones de Matlab. Si el ordenador local estuviese dotado de otra tarjeta de adquisición de datos y de otra planta de nivel, se podría realizar el control de las dos plantas simultáneamente haciendo uso de Remote EUP.

TÍTULO: **CONTROL REMOTO SOBRE LA PLANTA PEQUEÑA DE CONTROL DE NIVEL DEL LABORATORIO**

---

# ANEXOS

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBREIRO, S/N**

**15405 - FERROL**

FECHA: JUNIO DE 2017

AUTOR: EL ALUMNO

Fdo.: DANIEL MÉNDEZ BUSTO

# Índice del documento ANEXOS

# 9 Documentación de partida

Se presenta la adjudicación del Trabajo Fin de Grado.


# ESCUELA UNIVERSITARIA POLITÉCNICA

## ASIGNACIÓN DE TRABAJO FIN DE GRADO

**En virtud de la solicitud efectuada por:**
*En virtud da solicitude efectuada por:*

***APELLIDOS, NOMBRE:*** *Méndez Busto,Daniel*
*APELIDOS E NOME:*

**DNI:** [redacted] **Fecha de Solicitud:** Feb2017
*DNI:* *Fecha de Solicitude:*

**Alumno de esta escuela en la titulación de Grado en Ingeniería en Electrónica Industrial y Automática, se le comunica que la Comisión de Proyectos ha decidido asignarle el siguiente Trabajo Fin de Grado:**

*O alumno de esta escola na titulación de Grado en Enxeñería en Electrónica Industrial e Automática, comunícaselle que a Comisión de Proxectos ha decidido asignarlle o seguinte Traballo Fin de Grado:*

***Título T.F.G:***Control remoto sobre la planta pequeña de control de nivel del laboratorio

***Número TFG:*** *770G01A123*

***TUTOR:***(Titor) *Calvo Rolle,Jose Luis*

***COTUTOR/CODIRECTOR:*** *Oscar Fontenla Romero*

***La descripción y objetivos del Trabajo son los que figuran en el reverso de este documento****:*

A descrición e obxectivos do proxecto son os que figuran no reverso deste documento.

*Ferrol a Domingo, 11 de Junio del 2017*

Retirei o meu Traballo Fin de Grado o dia _____ de _____ do ano _____

*Fdo: Méndez Busto,Daniel*

***DESCRIPCIÓN Y OBJETIVO:***OBJETO:
El objetivo principal del presente trabajo es llevar a cabo el control remoto de la planta de nivel del laboratorio haciendo uso del entorno EASY JAVA SIMULATIONS y MATLAB. Se podrá comandar el sistema de forma remota desde cualquier lugar con un simple acceso a internet.

ALCANCE:
Los pasos que se siguen para la realización de este proyecto son los siguientes:
- Estudio y comprensión del entorno de programación EASY JAVA SIMULATIONS.
- Estudio específico para el caso de estudio de la programación en Matlab.
- Realizar la conexión entre EASY JAVA SIMULATIONS y MATLAB.
- Programar la interface en EASY JAVA SIMULATIONS.
- Pruebas y validación de la propuesta sobre la planta de laboratorio.
- Estudio comparativo de diferentes alternativas y propuestas de ampliación.
- Análisis de resultados y extracción de conclusiones.

# 10 Cálculos

Tras estudiar la respuesta del método de Relay - Feedback de la figura 8.5.0.6 se obtienen los siguientes parámetros:

| Parámetro | Valor |
|---|---|
| $T_c$ | 30 |
| $a$ | 32 |

**Tabla 10.0.0.1 –** Parámetros obtenidos con el método de Relay - Feedback

Haciendo uso de la ecuación 7.2.3.1, se halla el parámetro $K_c$.

$$K_c = \frac{4 * 50}{\pi * \sqrt{32^2 - 10^2}} = 2{,}094$$

Mediante el uso de la ecuación 7.2.4.1, se obtienen los parámetros del regulador correspondientes a la aproximación de Ziegler - Nichols.

| Parámetro | Valor |
|---|---|
| $k$ | 1,25 |
| $Ti$ | 15 |
| $Td$ | 3,75 |

**Tabla 10.0.0.2 –** Valor de los parámetros del regulador mediante Ziegler - Nichols

Los valores de los parámetros de la aproximación de Ziegler - Nichols No Overshoot se obtienen a partir de la ecuación 7.2.4.3.

| Parámetro | Valor |
|---|---|
| $k$ | 0,42 |
| $Ti$ | 30 |
| $Td$ | 10 |

**Tabla 10.0.0.3 –** Valor de los parámetros del regulador mediante Ziegler - Nichols No Overshoot

Haciendo uso de la ecuación 7.2.4.2 se obtienen los parámetros mediante Ziegler - Nichols Some Overshoot.

| Parámetro | Valor |
|---|---|
| $k$ | 0,69 |
| $Ti$ | 15 |
| $Td$ | 10 |

**Tabla 10.0.0.4 –** Valor de los parámetros del regulador mediante Ziegler - Nichols Some Overshoot

Los parámetros correspondientes a la aproximación de Tyreus - Luyben se calculan a partir de la ecuación 7.2.4.4.

| Parámetro | Valor |
|---|---|
| $k$ | 0,94 |
| $Ti$ | 66 |
| $Td$ | 4,76 |

**Tabla 10.0.0.5 –** Valor de los parámetros del regulador mediante Tyreus - Luyben

# 11 Otros Anexos

Se muestran los *scripts* de comunicación de la tarjeta de adquisición de datos NI USB-6008.

## 11.1. DAQ_Start [5]

**Código 11.1:** DAQ_Start

```
function []=DAQ_Start()
% This function configures the DAQ input and output
global AI
global AO

Device=daqhwinfo('nidaq'); % Find the desired device
ID=Device.InstalledBoardIds{1}; % Select the correct device from a list

AI=analoginput('nidaq',ID); % Inicialized of the input channel
AO=analogoutput('nidaq',ID); % Inicialized of the output channel

Input_channel=addchannel(AI,0); % Configure input channel (AI-0)
Input_channel.InputRange=[-10 10]; % Voltage input range
Input_channel.UnitsRange=[-100 100]; % Internal input range
Input_channel.Units='%'; % Internal units for input

Output_channel=addchannel(AO,[0 1]); % Configure output channel [AO-0 AO-1]
Output_channel.OutputRange=[0 5]; % Voltage output range
Output_channel.UnitsRange=[0 100]; % Internal output range
Output_channel.Units='%'; % Internal units for output

end
```

## 11.2. DAQ_Write [5]

**Código 11.2:** DAQ_Write

```
function []=DAQ_Write(Output_data_0,Output_data_1)
% This function write de desired value in the configured DAQ output

global AO

% Ensure that the value is inside the operation range
if Output_data_0>100
Output_data_0=100;
elseif Output_data_0<0
Output_data_0=0;
end

if Output_data_1>100
Output_data_1=100;
elseif Output_data_1<0
Output_data_1=0;
end

putsample(AO,[Output_data_0 Output_data_1]);

end
```

## 11.3. DAQ_Read [5]

**Código 11.3:** DAQ_Read

```
function [Input_data]=DAQ_Read()
% This function read the value in the configured input in the DAQ

global AI

Input_data=getsample(AI);

end
```

## 11.4. DAQ_Stop [5]

**Código 11.4:** DAQ_Stop

```
function [] = DAQ_Stop()
% This function is used to finaliced the uses of the DAQ

global AI
global AO

% Clear the input channel
stop(AI);
delete(AI);

% Clear the output channel
putsample(AO,[0 0]); % Ensure the output channel is resetted
stop(AO);
delete(AO);

end
```

TÍTULO: **CONTROL REMOTO SOBRE LA PLANTA PEQUEÑA DE CONTROL DE NIVEL DEL LABORATORIO**

---

# PLANOS

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBREIRO, S/N**

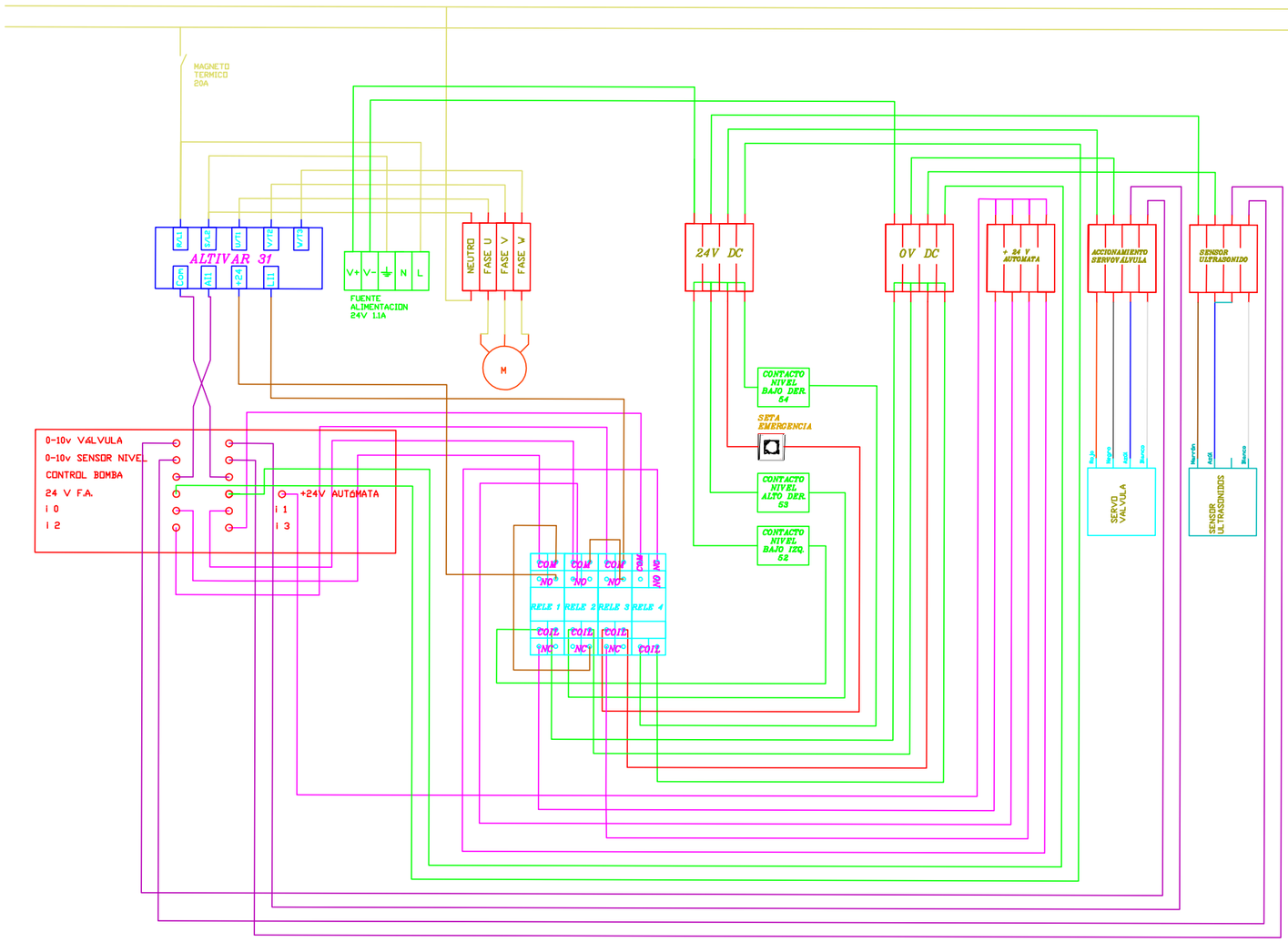
**15405 - FERROL**

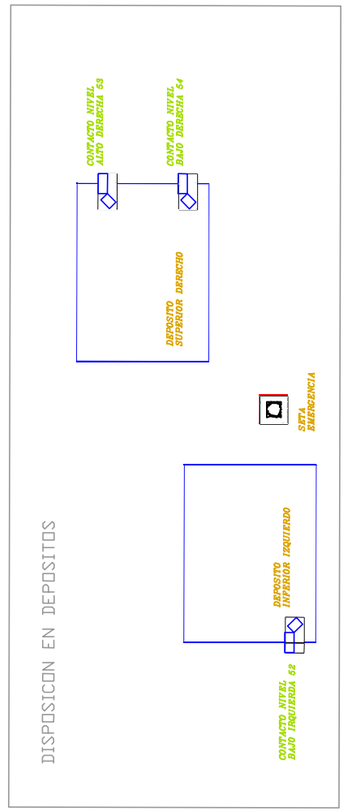
FECHA: JUNIO DE 2017

AUTOR: EL ALUMNO

Fdo.: DANIEL MÉNDEZ BUSTO

# Índice de planos




Nota: la posición relativa de las fichas de conexionado (en color rojo) se corresponde con la disposición final de ejecución.

| | |
|---|---|
| UNIVERSIDADE DA CORUÑA — ESCUELA UNIVERSITARIA POLITÉCNICA<br>GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA | TFG Nº: 770G01A108 |
| TÍTULO DEL TFG:<br>ESTUDIO DE MÉTODOS DE DETECCIÓN DE FALLOS PARA SISTEMAS SISO | |
| TÍTULO DEL PLANO:<br>ESQUEMA ELÉCTRICO PLANTA | FECHA: FEBRERO-2017<br>ESCALA: N/A |
| AUTOR:<br>FRANCISCO GARCÍA NOVO — FIRMA: | PLANO Nº: 01 |

TÍTULO: **CONTROL REMOTO SOBRE LA PLANTA PEQUEÑA DE CONTROL DE NIVEL DEL LABORATORIO**

---

# PLIEGO DE CONDICIONES

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBREIRO, S/N**

**15405 - FERROL**

FECHA: JUNIO DE 2017

AUTOR: EL ALUMNO

Fdo.: DANIEL MÉNDEZ BUSTO

## Índice del documento PLIEGO DE CONDICIONES

# 12 Condiciones de Trabajo

Todas las pruebas del presente Trabajo Fin de Grado se han realizado en el laboratorio de optimización y control de la Escuela Universitaria Politécnica de Serantes.

Se han realizado en un entorno limpio y seco, a una temperatura ambiente de 22ºC.

El depósito inferior que surte de líquido al superior, siempre se ha encontrado al máximo de se capacidad para no dañar la integridad del sistema durante el periodo de pruebas. La planta de nivel se encuentra sobre una superficie nivelada y rugosa para evitar caída de líquido.

Toda prueba realizada que no cumpla estas condiciones de trabajo no será válida.

TÍTULO: **CONTROL REMOTO SOBRE LA PLANTA PEQUEÑA DE CONTROL DE NIVEL DEL LABORATORIO**

---

# ESTADO DE MEDICIONES

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBREIRO, S/N**

**15405 - FERROL**

FECHA: JUNIO DE 2017

AUTOR: EL ALUMNO

Fdo.: DANIEL MÉNDEZ BUSTO

# Índice del documento ESTADO DE MEDICIONES

# 13 Mano de Obra

El tiempo empleado para el desarrollo de la aplicación Remote Control se muestra en la siguiente tabla:

| Tarea | Tiempo (h) |
|---|---|
| Comunicación entre EJS y Matlab | 16 |
| Programación del script en EJS | 60 |
| Programación de la *interface* en EJS | 60 |
| Pruebas finales | 60 |
| **Total** | **196** |

**Tabla 13.0.0.1 –** Horas de programación para el desarrollo de Remote EUP

TÍTULO: **CONTROL REMOTO SOBRE LA PLANTA PEQUEÑA DE CONTROL DE NIVEL DEL LABORATORIO**

---

# PRESUPUESTO

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBREIRO, S/N**

**15405 - FERROL**

FECHA: JUNIO DE 2017

AUTOR: EL ALUMNO

Fdo.: DANIEL MÉNDEZ BUSTO

# Índice del documento PRESUPUESTO

# 14 Mano de Obra

Se calcula el precio de la mano de obra valorando la hora de trabajo de un licenciado en Grado en Ingeniería Electrónica Industrial y Automática en 30 $euros/h$.

| Tarea | Tiempo (h) | Coste hora (euros) | Total (euros) |
|---|---|---|---|
| Ejecución de Remote EUP | 196 | 30 | **5880** |

**Tabla 14.0.0.1 –** Coste de Mano de Obra.

# 15 Licencias software

Las licencias software necesarias para la realización de Remote EUP son las siguientes:

| Licencia | Coste (euros) |
|---|---|
| Matlab R2014b | 500 |
| Toolbox DAQ | 250 |
| Toolbox Control System | 250 |
| Easy Java Simulations | 0 |
| **Total** | **1000** |

**Tabla 15.0.0.1 –** Coste de Licencias Software.

# 16 Presupuesto

El coste total de la aplicación Remote EUP es:

| Producto | Coste (euros) |
|---|---|
| Mano de Obra | 5880 |
| Licencias Software | 1000 |
| **Base Imponible** | **6880** |
| **IVA 21 %** | **1444.8** |
| **Total** | **8324.8** |

**Tabla 16.0.0.1 –** Coste de Remote EUP.

TÍTULO: **CONTROL REMOTO SOBRE LA PLANTA PEQUEÑA DE CONTROL DE NIVEL DEL LABORATORIO**

---

# ESTUDIOS CON ENTIDAD PROPIA

---

PETICIONARIO: **ESCUELA UNIVERSITARIA POLITÉCNICA**

**AVDA. 19 DE FEBREIRO, S/N**

**15405 - FERROL**

FECHA: JUNIO DE 2017

AUTOR: EL ALUMNO

Fdo.: DANIEL MÉNDEZ BUSTO

## Índice del documento ESTUDIOS CON ENTIDAD PROPIA

En este Trabajo Fin de Grado no se adjuntan estudios con entidad propia.